A 1Mb mixed-precision quantized encoder for image classification and patch-based compression

Van Thien Nguyen, William Guicquero, Gilles Sicard

A 1Mb mixed-precision quantized encoder for image classification and patch-based compression

Van Thien Nguyen, William Guicquero, and Gilles Sicard

***Abstract*—Even if Application-Specific Integrated Circuits (ASIC) have proven to be a relevant choice for integrating inference at the edge, they are often limited in terms of applicability. In this paper, we demonstrate that an ASIC neural network accelerator dedicated to image processing can be applied to multiple tasks of different levels: image classification and compression, while requiring a very limited hardware. The key component is a reconfigurable, mixed-precision (3b/2b/1b) encoder that takes advantage of proper weight and activation quantizations combined with convolutional layer structural pruning to lower hardware-related constraints (memory and computing). We introduce an automatic adaptation of linear symmetric quantizer scaling factors to perform quantized levels equalization, aiming at stabilizing quinary and ternary weights training. In addition, a proposed layer-shared Bit-Shift Normalization significantly simplifies the implementation of the hardware-expensive Batch Normalization. For a specific configuration in which the encoder design only requires 1Mb, the classification accuracy reaches 87.5% on CIFAR-10. Besides, we also show that this quantized encoder can be used to compress image patch-by-patch while the reconstruction can performed remotely, by a dedicated full-frame decoder. This solution typically enables an end-to-end compression almost without any block artifacts, outperforming patch-based state-of-the-art techniques employing a patch-constant bitrate.**

***Index Terms*—hardware-algorithm co-design, quantization, pruning, autoencoder, patch-based image compression**

I. Introduction and context

TWO main branches of research are pushing forward Deep Neural Networks (DNNs) into smart embedded systems and mobile devices [1]. Indeed, we can state that there could be antagonist objectives depending on the targeted applications (level of versatility and programmability), the available power consumption of the system, the complexity of the inference task and its required quality of service (*e.g.* level of accuracy).

The first branch aims at developing generic hardware platforms dedicated to multi-purpose DNNs with the highest TOPS/W as the main Figure of Merit. Note that, lots of works have been done in that direction. For instance, we can non-exhaustively mention the possible simplification of the expensive matrix-vector multiplication task, using either FFT-based computing [2], Strassen [3] or Winograd [4] algorithms. Recent works improve the performance and energy efficiency of accelerators by focusing on three main key points: reconfigurability, sparsity and weights bitwidth. Within the scope of a reconfigurable platform, we can cite [5], a deep neural architecture enabling highly reconfigurable patterns while targeting a wide range of models with a limited power budget of 479mW. Some other works also introduce specific architectures that limit data movement in CNN. [6] proposed Eyeriss - a spatial architecture with row-stationary dataflow that takes advantage of local data reuse mechanism even exhibiting a lower overall chip power consumption of 278mW. On the other hand, [7] handles sparsity with binary maps and arrays for nonzero values, skipping null activations to reach a power consumption of only 155mW. Variable bitwidth computations allow good trade-offs between accuracy and memory/power budgets as depicted in [8] that presents a DNN accelerator supporting variable weights bitwidth precision from 1 to 16 bits, with a power consumption ranging from 3.2mW to 297mW (depending on the master clock frequency and power supply). [9] proposed a computing-in-memory neural network processor efficiently dealing with sparsity and weight storage on SRAM. [10] presents a resource-efficient architecture for BNN inference accelerator, processing blocks in an output-oriented manner while skipping redundant operations.

The second branch is more related to ASICs (Application-Specific Integrated Circuits) that are dedicated to only certain tasks, considering both the hardware specifications and the algorithm perspectives proposing co-optimized designs. This promotes compact network topology design with improved power consumption efficiency while capping algorithmic loss of accuracy. For instance, [11] proposed an accelerator optimized for binary-weights CNN with a power consumption of 895µW. Using quantization technique in both analog and digital domain, Kim et al. [12] proposed an always-on face recognition processor integrated with a CMOS image sensor (CIS) consuming about 620µJ per inference. [13] even proposed a sub-10µW QQVGA imager enabling both motion detection with background estimation and face recognition using XOR-based edge extraction combined with a SVM. [14] proposed a QQVGA imager which can operate on convolution mode with ternary-weighted filters and Haar-like detection mode, under less than 206µW. [15] presented a CNN-based accelerator that can serve for both face detection and facial landmarks localization. More recently, [16] presented a digital Binary Neural Network (BNN) chip achieving a power consumption of 5.6mW. Low-precision CNN processors were also applied to real-time object detection task in [17] and [18]. In particular, [18] adopted a mixed data flow for each layer along with an intra-layer mixed weights precision quantization, in which the weights were decomposed to a dense binary kernel and a sparse 8-bit kernel in order to improve the accuracy/compression ratio trade-offs.

Van Thien Nguyen, William Guicquero, and Gilles Sicard are with the CEA-LETI, University Grenoble Alpes, F-38000 Grenoble, France (e-mail: vanthien.nguyen@cea.fr, william.guicquero@cea.fr, gilles.sicard@cea.fr).

Please refer to [19] for a more exhaustive review of published works on DNN accelerators (reporting their relative energy efficiency in terms of TOPS/W so as the types of quantization used and available operands).

Motivations:
Taking into consideration ASIC design issues, the work presented in this paper has a dual purpose:

- increase the application-versatility of an Image and Signal Processor, dedicated to classification and compression;
- improve the hardware efficiency and algorithmic performances trade-off (*i.e.* inference/reconstruction accuracy).

To this end, we propose a hardware-compliant mixed-precision encoder and its decoder counterpart. The main advantage of the proposed encoder topology is that it can be declined for both HW-light embedded inference (Figure 1, a) path) and image compression tasks (Figure 1, b) path). Indeed, the mixed-precision quantized encoder outputs discriminant patterns as a latent binary representation of the data. From this binary coding, we can thus either use directly a classifier to perform the embedded inference or a remote decoder network (PURENET) for image recovery (see Figure 2).

Contributions:

1) We introduce a mixed-precision encoder design with reconfigurability that may serve for both image compression and classification. It is noteworthy to mention here our novel adaptive quantization framework in which the quantizer is tuned according to the data during the training procedure such that the histogram of quantized levels is equally partitioned, namely histogram-equidistributed quantization. This framework is then applied it to the quinary and ternary weights in the encoder. Besides, the Batch Normalization obstacle is successfully replaced by a layer-shared Bit-Shift operation. We also propose a Half-wave Most-Significant-Bit (HWMSB) function for 2-bit activation with favorable hardware compatibility.
2) We propose a novel decoder called PURENET that takes as input the patch-based binary measurements from the quantized encoder. To our knowledge, this work presents one of the first low-precision quantized encoder for patch-based image compression. Our experiments show that image decompression can be performed without block artifacts at low bitrate.

The rest of this paper is divided as follows: Part II) presents related works, Part III) details our nonlinear quantized encoder design, including details of proposed algorithmic enablers. Part IV) describes the network topology of the Decoder for image reconstruction. Part V) presents simulation results on both image classification and image compression tasks. Two additional appendices are here to support our claims, reporting benchmarks regarding the effect of the encoder bottleneck and weight-related memory versus inference accuracy.

II. Related works

Weights and activations quantization, connectivity pruning and alternatives to the Fully Connected layer bottleneck are usually used as common techniques to facilitate the implementation of an Artificial Neural Network in terms of hardware mapping. Apart from algorithmic enablers for inference, the last part of this section depicts previous works on image reconstruction from patch-based compression.

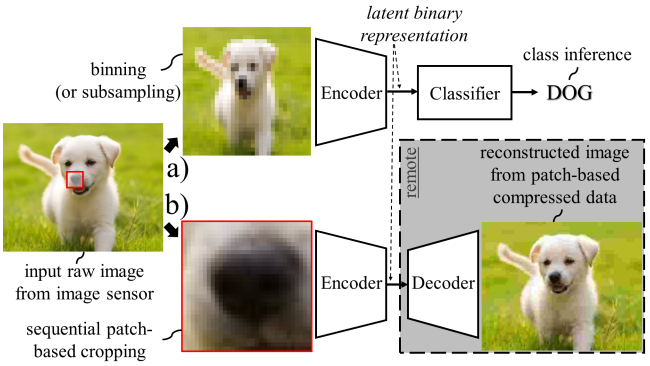

Fig. 1: The joint framework for both image classification and image embedded compression and remote decompression.

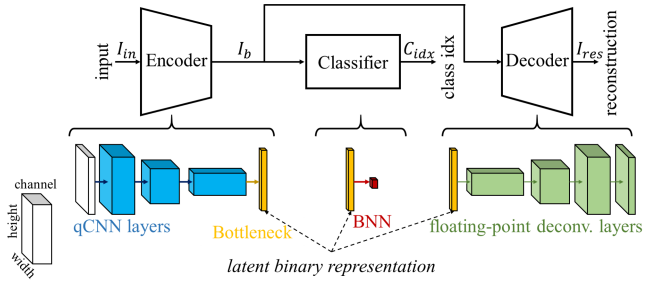

Fig. 2: Schematic description of our framework involving neural network topology parts.

A. Weights and activations quantization

Networks quantization compresses the model by reducing the bitwidth of weights and/or activations. The mapping from floating-point values to fixed values may be linear or logarithmic [20]. In Moons et al. [21], a generalized Q-bits quantization is proposed to study energy-accuracy trade-offs. In the most extreme case, networks can be trained with binary weights and activations as proposed in BNN [22]. For instance, $\{+1, -1\}$ matrix-to-matrix multiplications convert the power-costly MAC operations to the hardware-friendly XNOR and bitcount operations. XNOR-Net [23] namely gives rise to a scaling factor besides binary values to approximate activations/weights. In a more relaxing framework, Ternary Weights Networks (TWN) [24], [25] constraint the weights to $\{-1, 0, +1\}$ values. Although increasing 1 bit (≈ 0.6 bit in terms of entropy) for representing weights compared to the binary case, they allow zero values that attenuate the impacts of non discriminant neuronal activations. [26] shows that mixed precision DNNs with learnable parametrized quantizers outperform DNNs with fixed homogeneous quantizers. However, such a learnable quantizer will not be in the scope of this work due to its hardware-related requirements. The

common point of all the aforementioned techniques is that the input of the first layer and the output of the last layer are not quantized, as they play a key role in network's performance.

Quantized step in symmetric linear quantization: Although there exists several works on proposing non-linear quantization schemes, the hardware feasibility of those quantizers is still questionable. The linear symmetric quantization, on the contrary, may be easily adapted to an inference at the edge. However, the uniform step in this case is a crucial parameter that may jeopardize the model's performance if it is not properly chosen. Ternary Weight Networks [24] choose a step of $\Delta = 0.7\frac{\sum_{i=1}^{n}|W_i|}{n}$ where the mean-absolute norm factor of 0.7 is chosen assuming the weights distribution is uniform or gaussian. Learning the quantizer is also an active approach for this subject, for example the recent works of [27], [28] and [29]. In a different way from these works, in this paper, we propose to automatically adapt the quantized step based on layer-wise weights histogram equalization, in order to approximately maximize the entropy of quantized weights.

Batch Normalization in QNNs: An important remark about QNNs is that they generally fail to properly converge without Batch Normalization (BN) [30]. However this property becomes an obstacle for efficiently deploying QNNs in embedded systems, because the affine transform of BN at inference stage is still an expensive operation from the hardware point of view. Due to the crucial role of BN during the training process, there is no surprise that previous works either kept BN in full precision or avoided to use it. [31] proposed to fuse the batch normalization parameters into the kernel and bias of BNNs and perform an Addition with the 9b equivalent bias. Riptide [32] even approximated the variance of the BN and the scale terms of XNOR-Nets by bitshift operations, and combined them to embed both the magnitudes of weights and normalizations. Another scheme is Sign Comparison which absorbs the BN into the Sign function by an equivalent comparison between the pre-activation and the bias obtained after training [33], [34]. However, both of these two approaches still kept the presence of the bias which surely introduced an additional module in terms of hardware implementation. Besides, these specific schemes can only be applied in the case of activation binarization, otherwise in the case where BN is followed by a multi-bit quantization, they are no more applicable.

In our work, we also have a mixed-precision approach in order to improve the harware/algorithm trade-offs. Unlike [18] which applied an intra-layer mixed precision, we apply a layer-homogeneous quantization where we dedicate multi-bit quantization for weights and activations of the very first layers. The quantization process for activations should enable a hardware-friendly implementation mapping. Additionally, BN alternatives should not degrade network's performance so much, *e.g.* smaller than 1% compared to standard BN. Based on these requirements, three elements will be introduced: a generic framework for multi-bit quantization based on histogram which is applied to the case of quinary and ternary weights in our encoder; a Most-Significant-Bit (MSB) function for activations quantization that extracts the relative position of the most significant bit; and finally a BitShift-based Normalization (BSN) that approximates the affine transform of the whole BN layer at testing time by a single power-of-2 rescaling, without the introduction of additional biases.

B. Inter-layer connectivity pruning

Network pruning aims at reducing the number of operations, operands and weights in models along with overfitting, based on the assumption that fully connected layers –once trained– exhibit sparsity and redundancy. Han et al. [35] proposed a three-step weights pruning strategy including connection training, threshold-based weight pruning and retraining. However, a magnitude-based pruning strategy may result in irregular structures increasing hardware complexity to be properly taken into account. Therefore, several works focus on structural pruning with regularizer during training. Examples include channel-pruning [36], structural-aware pruning [37] and budget-aware training [38]. In particular, [39] introduces a novel hardware-compliant pruning framework where the weights are pruned within the weights-fetching groups. Besides the sparsity learning approach, [40] shows that a pre-defined random pruning can preserve the accuracy of its unpruned counterpart due to the DNN plasticity.

In addition to conventional pruning methods mentioned above, we can also cite some works aiming to reduce the cost of standard convolutions. MobileNet [41] introduced depthwise separable convolution, *i.e.* a single filter per input channel, followed by a regular 1×1 conv layer. ShuffleNet [42] simplified the fully-connected pointwise convolution in a Group-wise manner and a channel shuffle to equidistribute information and enhance correlation between channels.

In our work, we will similarly adopt a pre-defined structural pruning technique to reduce complexity of convolution layers (memory and computations) in the context of a quantized CNN. While ShuffleNet applies Group Convolution to 1×1 kernels, we propose to directly perform Group Convolution to 3×3 kernels on top of the convolutional layers.

C. Alternatives to the Fully Connected layer (CNN bottleneck)

Traditional deep CNN architectures consist of a convolutional block followed by one or many Fully Connected (FC) layers to perform the classification. Although it has shown the success on various datasets, these FC layers, in particular the very first FC layer (CNN bottleneck), often hold a huge percentage of model's parameters. For resource-constrained applications, this becomes a main limitation for the CNN deployment. Besides, these FC layers are also a factor that is prone to overfitting. There exists several methods replacing the first FC layer to achieve a better efficiency. Global average pooling is first introduced in [43] that outputs only one value per feature map. In the same perspective, [44] proposed a 3D convolution in which only local feature maps are combined to get the prediction for each class. Both these two approaches dramatically reduces the number of parameters while achieving a competitive performance compared to FC-based classifier. They all demonstrate that the FC layers exhibit redundancy and can be replaced with an acceptable loss of accuracy and generalization of the model. Indeed, both global average pooling and 3D convolutions are just a formulation of

linear transformation with reduced support size compared to a dense support of FC. However, while the difference between these classifiers and FC-based classifiers is not significant in the context of full-precision models, there may be a clear gap in the specific case of quantized neural networks.

Another direction to reduce the number of parameters of FC layers is to use a fixed transform whose weight parameters are predefined. The deployment of such a model with fixed parameters is more suitable for devices with limited resources. [45] shows that any fixed orthogonal matrix can be used to efficiently replace a learnable FC. Even if it does not reduce the total number of operations, they demonstrate that a Hadamard matrix can improve the efficiency in terms of hardware implementation. Besides, the use of pseudo-random deterministic projections can also be a good alternative because they preserve the linear separability while being hardware-implementable, this without additional memory needs [46].

In this work, we propose an alternative to the first FC layer with input data of shape $H \times W \times C$ and output of C hidden units. Our idea basically consists in using a depthwise convolution (DWConv) of kernel size $H \times W$ followed by an FC layer without activation in-between. Later experiments show that this approach achieves highly competitive results.

D. Autoencoder for patch-based image compression

Patch-based (or block-based) encoding is preferred to its full-resolution counterpart, as it reduces the processing complexity and storage-related costs. Besides standards such as JPEG [47], there exists a lot of other patch-based compression schemes, from dimensionality reduction using block-based Compressed Sensing (BCS) [48], [49] to deep learning-based approaches [50]. In the BCS framework, the high-dimensional image is divided into non-overlapping patches which are then projected separately onto a low-dimensional space via a common projection matrix, therefore we can consider it as a linear encoding scheme. The reconstruction of the entire image from these patch-based measurements can be done using iterative algorithms [51], [52] or recent learning-based methods [53], [54]. On the other hand, neural network-based algorithms have been used as an alternative approach for image compression. Toderici *et al.* firstly introduced a Long Short-Term Memory (LSTM)-based autoencoder [55] enabling a patch-based 32×32 thumbnails compression at variable compression rates, which encodes the residual error between the current reconstruction and the original image at each iteration. This framework was then developed and applied to full-resolution image compression [56] with the intensive use of an LSTM-based entropy coder to capture long-term dependencies of patches. Having the same perspective, [57] introduced a progressive encoding scheme exploiting dependencies between adjacent patches. Although achieving favorable results at shallow compression rates, these Recurrent Neural Network (RNN)-based autoencoders are not suitable for being deployed in resource-constrained systems due to their complexity and hardware incompatibility. Also, these methods are not purely patch-based as they make use of spatial coherence between adjacent regions to –more efficiently– encode every patch and alleviate the block artifacts. [58] and [59] proposed deep learning image compression techniques while adapting the region-based bitrate accordingly to the local content of image. Note that even if such an adaptive bit allocation would improve the overall performances, it does not fit within our hardware-restricted and patch-by-patch compression framework. Ensemble learning is recently deployed in [60] which uses several networks of the same structure but different parameters, and a network is selected when encoding each image block before signaling to the decoder.

QNNs for image compression: Quantizing the weights and activations has been showing excellent results on semantic tasks such as image classification and object detection. However, representing the weights and activations using a low precision causes a considerable loss of information at pixel level, which prevents QNNs from achieving good performances for image compression or super resolution. This explains the absence of quantized models for these pixel-level tasks in the related state-of-the-art. We can only cite some works of [61] and [62] that are dedicated to super resolution. In the case of image compression, the role of pixel-level information is more crucial in encoding image data, hence there exists hardly no prior works on quantized models for this task. However, in this paper, we use the same quantized encoder topology for patch-based image compression as the one designed for image classification. We show that even with such an encoder, the reconstruction is still guaranteed by the proposed PURENET decoder with almost no block artifacts at the equivalent bitrate of 0.25 bits-per-pixel (bpp).

III. NONLINEAR QUANTIZED ENCODER

Our proposed framework first targets inference for always-on embedded systems. To ease fair comparisons with state-of-the-art approaches, we use the basic but reproducible CIFAR-10 image classification dataset [63]. Even though a practical application may differ for final sizing of the topology, CIFAR-10 is generally deployed to benchmark both algorithm and hardware designs [31], [33], [10], [34], [64]. Based on literature reviews related to network quantization and the VGG model [65], we took the VGG-7 model for the sake of simplicity as the pivotal element for constructing our topology variants. Note that all the reported contributions here are compatible with other kinds of neural networks architectures such as ResNet [66], MobileNet [67], ...

Figure 3 depicts our hand-crafted mixed-precision network topology integrating all proposed algorithmic enablers for the image classification task. The model is divided into 5 main modules, with 3 Convolution blocks, 1 Bottleneck layer and 1 output Classifier. In this specific setting (*i.e.* the baseline model), a multi-bit quantization scheme for both activations and weights is already applied to the first two convolutional modules, *i.e.* Quinary weights Quantization (QQ) for the first, Ternary weights Quantization (TQ) for the second and Binary weights Quantization (BQ) for all the rest. Since the first conv layers are crucial to extract meaningful information therefore more bits for them leads the network to better performance trade-offs. Note that the very first conv layer additionally embeds channel-wise biases to properly enable image dynamics

TABLE I: NQE topology with details on layer inputs and kernels precision.

Layer		Input shape	Weight shape	Input precision	Weight precision
Conv layer		$32 \times 32 \times 3$	$3 \times 3 \times 3 \times F$	8	3
Conv layer 2		$32 \times 32 \times F$	$3 \times 3 \times F \times F$	1	3
Conv layer 3		$16 \times 16 \times F$	$3 \times 3 \times F \times 2F$	2	2
Conv layer 4		$16 \times 16 \times 2F$	$3 \times 3 \times 2F \times 2F$	1	2
Conv layer 5		$8 \times 8 \times 2F$	$3 \times 3 \times 2F \times 4F$	2	1
Group Conv ($G = 4$)		$8 \times 8 \times 4F$	$3 \times 3 \times F \times 4F$	1	1
Bottleneck	Depthwise Conv	$4 \times 4 \times 4F$	$4 \times 4 \times 1 \times 4F$	1	1
	FC	$1 \times 1 \times 4F$	$4F \times 4F$		
FC		4F	$4F \times 10$	1	1

feature extraction. We also put HWMSB activations at the end of the first two convolutional blocks and the heaviside activation at the end of the third block, as having zero values –instead of signed ones– helps learning to discriminate better data features. The first non-convolutional layer is the bottleneck which holds the half of model's parameters in its baseline format (*i.e.* a FC layer). This first FC is thus replaced by a depthwise convolution followed by a far smaller FC. These first fours blocks constitute together what is denoted a Nonlinear Quantized Encoder (NQE). The final block as for it, is a classifier composed of one single FC. From a top-level system view, the NQE will perform the image compression over non-overlapped 32×32 patches while in classification mode it is combined with the last-stage classifier. We also want to stress that applying 2×2 MaxPooling (MP2) on a binarized tensor results in a tensor with almost all ones, confusing the training procedure when choosing the argmax positions during backpropagation. However, applying MP2 over quantized values is more hardware-friendly than over full-precision values. In Figure 3, MP2 are thus put after HWMSB but before Heaviside keeping in mind that from the hardware point of view (*i.e.*only for feedforward pass) this last order can be reversed. Note that details of the layer weights and input configuration are described in Table I.

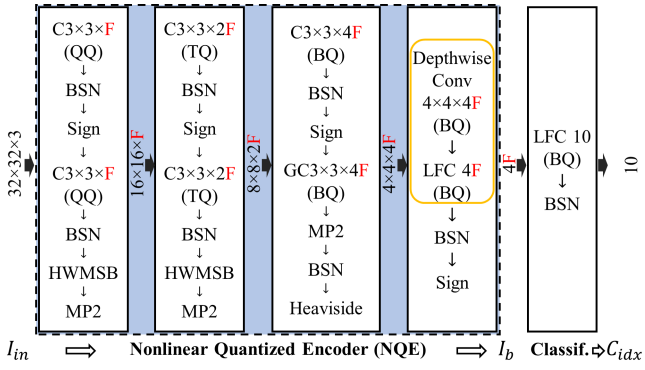

Fig. 3: Topology of the Nonlinear Quantized Encoder (NQE) + Classifier. F in red stands for the hyperparameter corresponding to the size scale of the feature map (*i.e.* the number of the feature map of the first convolution module). GC stands for Group-wise Convolution of 4 groups.

A. *Histogram bins equidistributed quantization*

A linear symmetric quantization of restricted range of odd n discrete values ($n > 2$) can be described as follows:

$$q(x;\Delta) = \frac{2}{n-1}\text{Clip}\left(\lfloor\frac{(n-2)x}{2\Delta}\rceil, \frac{1-n}{2}, \frac{n-1}{2}\right), \quad (1)$$

where the parameter Δ controls the quantization step and clipping which is defined as $\text{Clip}(x;a,b) = \min(\max(x,a),b)$ with $a < b$. In this formulation, all quantized values $q \in \frac{2}{n-1}\{\frac{1-n}{2} + 1, ..., 0, ..., \frac{n-1}{2} - 1\}$ are uniformly distributed inside the interval $(-\Delta, \Delta)$, bounded to ± 1. Indeed, thanks to the $\frac{2}{n-1}$ scale factor, all the quantized values are shrunk in the interval $[-1, +1]$.

Figure 4 depicts the case of $n = 3$ (ternary quantization [24]) and $n = 5$ (quinary quantization). This formulation deeply depends on the quantization step parameter Δ, and how to choose an optimal value for it is still an open question. In the Ternary Weight Networks [24], $\Delta = \tau\frac{\sum_{i=1}^{n}|W_i|}{n}$ where the mean-absolute norm factor $\tau = 0.7$ is estimated by assuming the weights distribution to be either Uniform or Gaussian. We argue that in many cases, depending on the layer's position, the model topology, the training procedure and the inference task, these assumptions are usually invalid, resulting in a sub-optimality and an unbalance between the number of weights at each quantized level. For example, if the floating-point values concentrate mainly around zero, it is likely that there may be too many zeros over other quantized values.

Starting from this analysis, we propose to estimate the mean-absolute norm factor based on the histogram of the proxy weights, such that the quantized weights are distributed with nearly equi-probabilities. This way, we aim to increase the entropy of the quantized weights, and consequently, the diversity of the output values. Let us denote $q_0, q_1, ..., q_n$, $n+1$ points which divide the weights histogram into n equal quantiles, where q_0 and q_n are the minimum and maximum values, respectively. Observing that during the training procedure, the weights distribution may change, but the median value usually stays around zero, we assume that these quantiles are approximately symmetric. Concretely, to equidistribute the histogram bins of quantized weights, the parameter Δ and τ are re-estimated and updated according to the updated proxy weights after each training epoch such that these quantile points correlate with the thresholds in the quantization function. To update the proxy weights during each epoch, we keep using the STE [68] strategy with a clipped identity, while Δ

and τ remain fixed between mini-batches. For instance, in the case of ternary quantization, the points q_1 and q_2 in Figure 4 must match the points $-\Delta$ and Δ, therefore we have the approximation $\Delta = \frac{|q_1|+q_2}{2}$. Similarly, in the case of quinary quantization, the points q_1, q_2, q_3, q_4 should coincide with the points $-\Delta, -\frac{\Delta}{3}, \frac{\Delta}{3}, \Delta$, therefore we have the approximation $\Delta = \frac{3(|q_1|+|q_2|+q_3+q_4)}{8}$. From this approximated Δ, we can thus estimate an equivalent mean-absolute norm factor τ (cf. [24]) using the following equation:

$$\tau = \frac{n\Delta}{\sum_{i=1}^{n} |W_i|}. \tag{2}$$

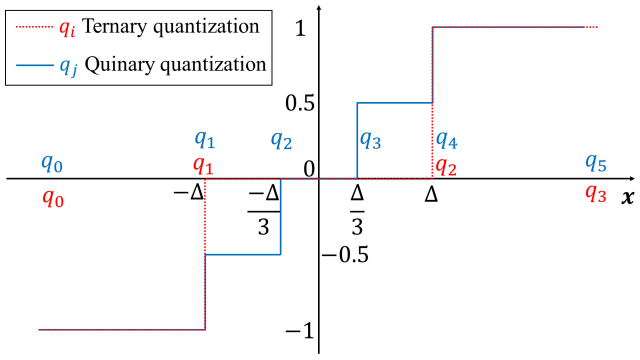

Fig. 4: Symmetric linear quantization with histogram bin equidistribution. The ternary quantization is represented with red dotted curve and red quantiles q_i. The quinary quantization is represented with blue solid curve and blue quantile q_j. For simplicity, we choose the same Δ for both two quantizations, but in practice, they usually have a different estimated Δ.

While the ternary weight networks are widely deployed in designing efficient CNNs, the quinary quantization is still quite novel. Even having only 5 over 8 quantized values possible of a 3-bit representation, a kernel of $0, \pm 0.5, \pm 1$ greatly simplifies the matrix-matrix multiplication into logical operations, bitshifts, and accumulations. It also limits the number of bits to store the intermediate values, while increasing significantly the representability of the kernels. Therefore, we apply quinarization and ternarization to the first two convolution modules only, in order to extract more meaningful information from image features while limiting the overall impact on memory needs.

B. Half-Wave Most-Significant-Bit (HWMSB) activation

To compensate additional hardware needs when using more bits for intermediate values, we propose to simply extract the position of the most significant bit of the income value. This operation advantageously embeds two wanted features, namely log_2 dynamic range compression and intrinsic requantization. Its original real-valued function mapped in the $[-1, +1]$ range can be defined as:

$$f(x) = \begin{cases} \text{sign}(x)\ \min\left(\frac{4+\log_2(|x|)}{3}, 1\right) & \text{if } x \geq \frac{1}{8}, \\ \frac{8x}{3} & \text{otherwise.} \end{cases} \tag{3}$$

The MSB quantization function and its Straight-Through-Estimated (STE [68]) gradient are then described as follow:

$$q(x) = \begin{cases} \text{sign}(x)\ \min\left(\frac{\lfloor 4+\log_2(|x|)\rfloor}{3}, 1\right) & \text{if } |x| \geq \frac{1}{8}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

$$\frac{\partial q(x)}{\partial x} = \begin{cases} \frac{1}{3|x|\ log2} & \text{if } \frac{1}{8} \leq |x| \leq 1, \\ \frac{8}{3} & \text{if } |x| < \frac{1}{8}. \end{cases} \tag{5}$$

When this MSB is followed by a ReLU activation, only positive values can be passed and negative values are zeroed out. We call this combination as Half-wave Most-Significant-Bit (HWMSB) activation. Unlike the MSB function, the HWMSB activation has only 4 possible output values $\{0, 1/3, 2/3, 1\}$, therefore it needs only 2 bits to represents the output data. Another advantage of HWMSB compared to MSB is that attenuating negative values improves the learning as ReLU does in several topologies. Table II reports the value mapping between the decimal, naive binary representations and the outputs. The first significant bit is assigned to the third bit on the right of the point. We call this the reference position, which is determined by the integer bias of 4. If the input is multiplied by a power-of-2 factor before the MSB, we can absorb this multiplication into the MSB by just shifting the reference position accordingly. Note that this quantization scheme differs from existing logarithmic quantizations such as [20] on these two points, first it has the integer bias of 4 and it also zeroes out negative values. Note that the normalization factor (here, 3 to keep the dynamic in the wanted range) can be assigned independently to this scheme.

TABLE II: 2-bit HWMSB input-output mapping with naive binary representation (sign+magnitude)

Input		**Output**	
Decimal	***Binary***	***Decimal***	***Binary***
$x < 0.125$	x.000xx/ 1.xxxxx	0	0.00
$0.125 \leq x < 0.25$	0.001xx	1/3	0.01
$0.25 \leq x < 0.5$	0.01xxx	2/3	0.10
$x \geq 0.5$	0.1xxxx	3/3	0.11

C. Layer-shared BitShift-based Normalization (BSN)

BN keeps a crucial role to QNNs, especially BNNs, as these networks fail to well converge without a proper rescaling. However, BN is not tractable for Deep Learning in highly constrained embedded systems, since at inference time, it consists of one full-precision addition and multiplication per scalar, which is a computational-demanding operation. A quantized BN from scratch is definitely not as robust as the standard BN as it is difficult to estimate the appropriate scaling factor, and unfortunately involves a significantly lower network's accuracy (*e.g.* much larger than 1%).

BN replacement by a single BitShift: The straightforward option considered here is we still employ the standard BN to train the model from scratch and then simplify all BN affine transforms by a single BitShift approximation, with later retraining in order to update the weights accordingly. This two-step training procedure allows preserving its accuracy

performances with a high simplification of the final hardware implementation. We thus approximate the scale constants of BN layers obtained after first training in a power-of-2 fashion, that advantageously corresponds to the bitshift operation. After the training stage, BN layer has properly estimated batch statistics μ and σ^2, respectively representing the moving mean and the moving variance. Let us recall that at the inference stage, the BN consists in processing the input x to provide the output y as follows:

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \equiv \hat{\gamma} x + \hat{\beta} \tag{6}$$

where –using the same notations as in [30]– $\hat{\gamma} = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$ and $\hat{\beta} = \beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}$ are equivalent to a scale and an offset (*i.e.* channel-shared additional weight and bias if applied after 2D convolution, and unit-shared if applied after 1D Dense layer, therefore we have a vector of different $\hat{\gamma}$ for each BN). Concretely, in our framework, we choose the 0.9-quantile value from these scales at each layer, denoted as $\tilde{\gamma}$, to serve as the unique scale for all the BSNs, approximating it as follows:

$$y = 2^{\lfloor log_2 |\tilde{\gamma}| \rfloor} x \tag{7}$$

Note that the experimentally chosen 0.9-quantile is considered as it is a good trade-off between the maximum value that may explode the dynamic range along with the gradient, and the minimum value that may slow down the gradient update of previous layers. After replacing all BN layers of the model by this single BSN transform, we train again the network one more time. Note that for all the BSNs, we can get rid of the bias $\hat{\beta}$ as it does not improve the inference in general, at the expense of additional computations for hardware mapping.

Hardware implementation of BSN: Generally, having only one bitshift-based scale replacing the whole BN layer considerably reduces the computational complexity in all cases, regardless its following quantization. While needing only 4 bits to store the bitshift scale, all input data will share a common bitshift operation, that is much cheaper than affine transforms of different scales and biases, even if theses parameters are quantized. Moreover, in the context of our model topology where we have either HWMSB, Sign or Heaviside quantization after the BSN, this becomes more advantageous. Clearly, the single bitshift keeps the sign of data unchanged, therefore, it has no impact to the results of the later Sign or Heaviside quantization. Consequently, the BSNs followed by Sign or Heaviside quantization do not need to be explicitly implemented. Similarly, we can also get rid of the final BSN as it does not change the order of the logit predictions. In addition, the bitshift scale of BSN can be intrinsically fused into the HWMSB by just shifting the reference position of the HWMSB accordingly.

D. Pre-defined pruning with Group-wise Convolution (GC)

Inspired by ShuffleNet [42] and justified by the advantages of a structurally pre-defined pruning, we propose to perform convolutions of CNN in a Group-wise manner instead of an all-channel-fully-connected conventional topology (see Figure 5). This group-wise pruning is only applied to the last convolutional layer of the NQE, as it contains most of the parameters. A Group-Convolution reduces the parameters and the number of MACs by a factor equal to the number of groups. Consequently, it also reduces the memory needed for intermediate values. Compared to unstructural pruning scheme, a predefined structural pruning like the Group Convolution may be embedded directly to the hardware platform, without the need of additional memory to save the connection positions.

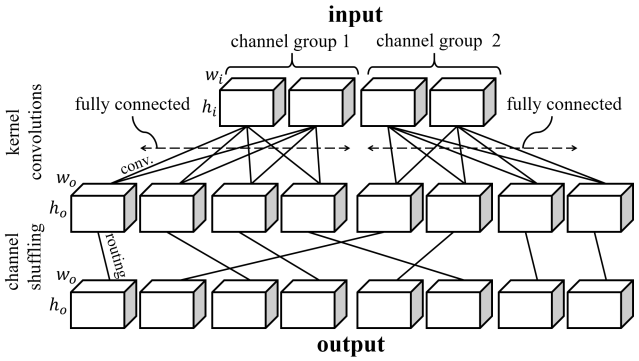

Fig. 5: Example of Group-wise convolution with a 4-channel input divided into 2 groups and 8-channel output. The intermediate values are also divided into two groups, and each convolution is performed with a kernel that takes only two input channels from the corresponding group. These output channels are then structurally shuffled.

E. Compression of the bottleneck dense layer

In our NQE, the first affine transform performed after conv modules has an input of shape $4 \times 4 \times 4F$ and outputs $4F$ hidden units. In the case of a dense layer, it will contain $256F^2$, *i.e.* almost 50% of the model's parameters. Even though the weights are binary, it still requires a large percentage of the total memory needs. Therefore, replacing this bottleneck layer while preserving the model's performance is crucial for improving the overall efficiency. In this paper, we propose to firstly use a depthwise convolution of kernel size 4×4, to transform the 3D tensor into a vector of length $4F$. Formally, this depthwise convolution is equivalent to a block-diagonally connected layer, with only $4 \times 4 \times 4F = 64F$ learnable parameters. Then we have a square learnable FC layer with size $4F$, therefore holding only $16F^2$ parameters. Since there are no activation between these two sub-layers, this approach is equivalent to decomposing the fully dense matrix into two sub-matrices, one for only spatial operation (depthwise convolution), the other for the combination of channels (dense layer). Consequently, the total number of parameters is $16F^2 + 64F$, which is very small compared to the initial $256F^2$ parameters of the FC version of the bottleneck. In Figure 3, these two layers are surrounded by a yellow rounded rectangle, denoting that they form together an alternative to the bottleneck dense layer.

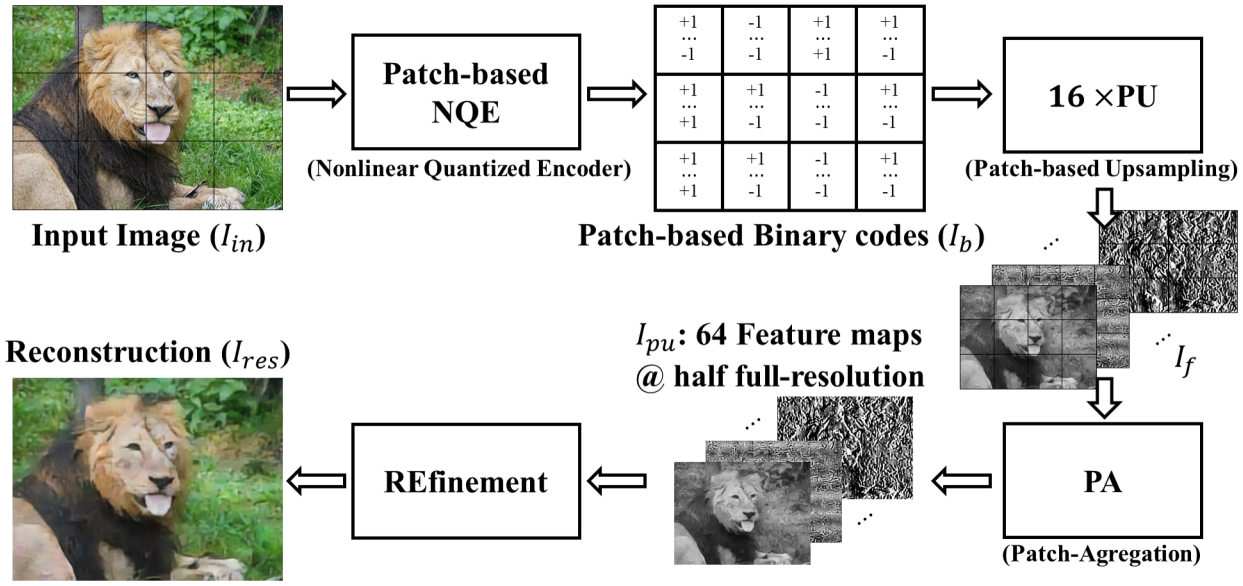

Fig. 6: The patch-based encoding with Full-Resolution PURENET decoder. Compressed binary codes are vector of length 256, corresponding to the bitrate of 0.25bpp. Note that without the Patches Aggregation (PA), this topology may still be applied to the decoding of patches independently. This variant without PA is denoted as patch-independent PURENET (PI-PURENET).

IV. Image reconstruction from patch-based quantized measurements with PURENET

Figure 6 shows the proposed combination of an image patch-based encoder with a full-frame decoder. The large image is first divided into small patches of size 32×32 and then processed by the NQE to obtain a binary representation for each patch independently. The Patch-based Upsamling (PU) module will learn to increase the spatial resolution of these codes from 1×1 (vector) to 16×16 (*i.e.* half of the final full-resolution). These patches are then aggregated together to form proxy feature maps at $\frac{1}{2} \times \frac{1}{2}$ full-resolution, which are then processed by the Refinement module to obtain the final reconstruction. The term PURENET hereafter denotes for the Patch-based Upsampling and REfinement NETwork.

Upsampling module: In details, the Upsampling model contains 4 blocks of Transpose Convolution + BN + ReLU (CBR). Each Transpose Convolution (ConvT) is of kernel size 3×3 and strides of 2. Figure 7 depicts the topology of this module. In PURENET, the Upsampling module is independently applied to every patch, so that the information of each patch is preserved and do not mix with the neighbors. We observe that aggregating the patches before the final 2×2 upsampling seems a good trade-off between alleviating the block artifacts and limiting the color errors due to the mixture of patches.

Refinement module: After the aggregation of all patches, we obtain a tensor of half resolution of the original image. The Refinement model in Figure 8 then allows smoothing the image, hence removing unwanted block-artifacts. Each patch is now reconstructed by using not only its own information but also its neighborhood. In particular, the Refinement model (see Figure 8) makes use of several Residual-Concatenation (RC) blocks before the final upsampling in order to reach the original full-resolution. Each RC block consists of 2

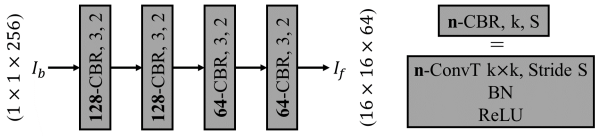

Fig. 7: PU: The Patch-based Upsampling performed using 4 CBRs (ConvT + BN + ReLU), all with a kernel size of 3×3 and a stride of 2.

CBRs, with a feature maps fusion inserted in-between that concatenates the RC input with the output of the first CBR and an add-skip connection at the output. Once the last upsampling is performed, the image feature maps pass through a last RC block followed by a self-attention mechanism. In this sub-module, each of the two branches contains a CBR with pointwise (1×1) convolution, and one branch has a softmax activation at the end to normalize the response of the pixels across the channel dimension. The two branches are then combined together by an element-wise multiplication, before outputting an RGB image for the final reconstruction.

PURENET training: We adapt a two-stage training procedure for PURENET. Firstly, we train the NQE with the patch-independent PURENET (*i.e.* PURENET without the Patches Aggregation, PI-PURENET) so that the NQE learns to compress 32×32 patches. After this stage, we obtain patch-based binary codes for each image which are then used as the input of PURENET in the second training stage. Finally, the pre-trained PI-PURENET is used to initialize the weights of corresponding modules (Patch-based Upsampling and Refinement) in the full-resolution mode (PURENET). This trick significantly accelerates the model convergence, also enhancing a bit the overall compression performance.

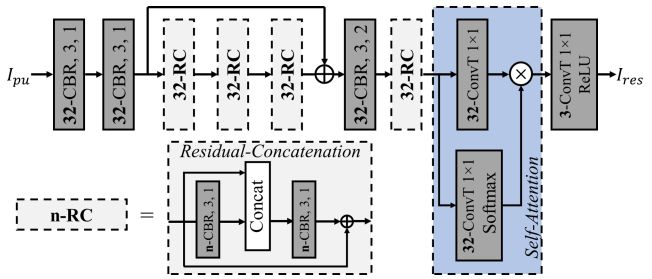

Fig. 8: RE: Refinement model architecture with Residual and Concatenation (RC) Block. The parameter $n = 32$ denotes the number of feature maps.

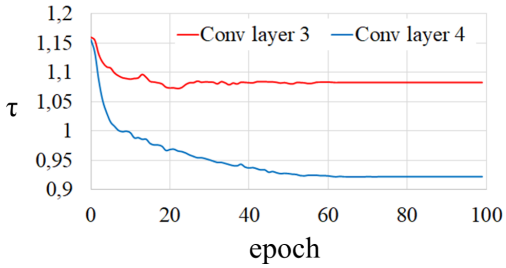

Fig. 9: τ evolution during training (quinary quantization).

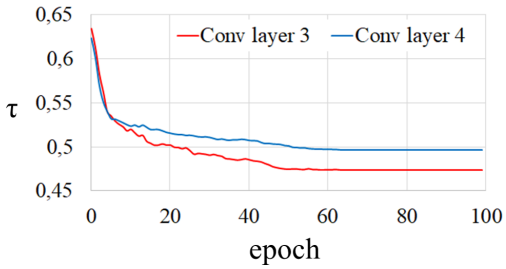

Fig. 10: τ evolution during training (ternary quantization).

V. Simulation Results

In this section, we target a configuration such that the memory budget for embedded classification is only approximately of 1Mb (with naive weights encoding), by choosing $F = 64$. Note that if we encode the quinary (2.32b) and ternary (1.58b) weights properly with an entropy coder, the on-chip memory may even be reduced to a sub-1Mb budget (which is not dealt in this paper). Note that all software implementations of this paper have been done in Python using a Tensorflow2 backend.

A. Image classification on CIFAR-10

In order to train our models, we apply the same data augmentation scheme as proposed in [69], *i.e.* a 4-sided 4-pixel padding followed by a random crop with random horizontal flip in order to provide 32×32 images. For classification task, the models are trained with a batch size of 50 using a squared hinge loss and the Adam optimizer [70]. 100 epochs are performed for the first training with standard BN and then 120 epochs for the fine-tuning stage, with BSN. The learning rate is initialized at 10^{-3} and decreased with an exponential decay with the rate of 0.8. The reported average accuracies (Table III) are over 3 realizations for each point.

Curve of τ during training: Figures 9 and 10 show the variation of the mean-absolute norm factor τ (defined in Equation 2) in the case of quinary (1st and 2nd Conv layer) and ternary quantization (3rd and 4th Conv layer). As can be seen on those figures, the mean-absolute norm factor decreases as the weights concentrate more around the zero point to finally converge when the learning rate becomes smaller and the model itself reaches a more stable state. The final values of the mean-absolute norm factors differ from each other, even for the same type of quantization. It is worth mentioning that in the case of second conv module (ternarization), the obtained factors (0.486, 0.498) appear to be much smaller than the value reported in [24] (*i.e.* 0.7).

Comparison with prior works: Table III provides a comparison with state-of-the-art CNN accelerators that have been recently demonstrated, mainly focusing on the input activation/weight precision, Batch Normalization implementation and the memory-accuracy trade-off. Both BNN and mixed-precision based designs are shown. For a relative complex dataset like CIFAR-10, a multi-bit quantization accelerator is proved to be more robust than a binarized accelerator. An important question has thus arisen: how to design such a mixed-precision architecture with higher accuracy, with lower memory needs and a simpler hardware implementation? While the two mixed-precision designs in [34] and [64] obtain an accuracy larger than 90%, they also contain on-chip memory of nearly 30 and 19Mb, which generally overpass the capacity of resource-constrained ASICs. On the other hand, BNN-based designs obtain lower accuracy at lower on-chip memory. Our NQE topology achieves 87.48% accuracy in average, while requiring approximately 1Mb of weight parameters. Compared to the reported design with the nearest accuracy [31], our design requires 2.4× less on-chip memory, with a 1.4% higher accuracy. Furthermore, the BSN offers a great relaxation for a future hardware implementation compared to the alternative approaches to handle normalization layers. Additional results that demonstrate the effectiveness of the proposed framework can be found in the related Appendix B.

B. Full-frame image compression

In this section we only focus on the reconstruction of images with a VGA resolution (480 × 640). To this end, each image is divided into 15 × 20 non-overlapping patches of 32 × 32 pixels to apply the NQE compression scheme.

VGA images dataset: We employ the DIV2K dataset [71] which includes 800 images for training and 100 validation images for testing. In order to meet the target resolution and because the original dataset provides a large variety of image resolutions, the images are cropped accordingly to the target height/width ratio and then resized with a Lanczos kernel with radius of 3. Then we extract all non-overlapping patches of size 32 × 32, obtaining 240k patches for the training of NQE.

NQE training: The NQE is trained with the PI-PURENET decoder using a batch size of 100, during 60 epochs with the standard BN, and then with 30 epochs after replacing BN by

TABLE III: Comparison of low-precision CNN processors (CIFAR-10 image classification task use case).

		Kim et al. [10] *TCS'21*	Valavi et al. [33]*JSSC'19*	Bankman et al. [31]*JSSC' 19*	Jia et al. [34] *JSSC' 20*	Cai et al. [64] *JSSC' 20*	***This work***
Weight precision		1	1	1	4	2	1,2,3
Input Activation precision		1	1	1	4	4	1,2
Batch Norm	Share Level	channel/unit	channel/unit	channel/unit	channel/unit	channel/unit	layer
	Bias-precision	6	6	9	6	10	0
	Implementation	Sign Comparison	Sign Comparison	Addition	Sign Comparison	Fused into Kernels	Intrinsic
On-chip Memory (Mb)		14.022	2.4	2.624	29.873	18.607	1.073
Accuracy (%)		88.80	84.37	86.05	92.70	90.03	87.48

BSN. To train this auto-encoder structure, we used the Mean Square Error (MSE) loss with the Adam optimizer.

PURENET training: The PURENET decoder is then trained (with a fixed NQE) in 30 epochs with a small batch size of 1 (mainly due to computational resources limitations). To obtain a proper convergence of the decoder, the learning rate is initialized at 0.001 and then rescaled by 0.95 at each epoch after the fifth epoch. We then freeze the Patch-based Upsampling module and fine-tune the Refinement with a batch size of 2 during 30 epochs. The learning rate is also initialized at 0.001 and then rescaled with the same factor 0.95 after the 10^{th} epoch.

Comparison with state-of-the-art methods: We compare NQE - PURENET with 4 other methods: JPEG [47] which is patch-based standard image compression technique (*i.e.* using an entropy coder after a sparsifying transform), one BCS encoding with regularization-based iterative method for decoding consisting in a L1-regularization on a 2D-Daubechies Wavelet Dictionary applied to 3D image gradients (denoted WD-TV3D, inspired from [52] and giving better results than a basic TV [72]; the RNN-based autoencoder [56] called Compressive AutoEncoder (CAE) dedicated to full-resolution image compression; our NQE with PURENET for patches reconstruction, namely NQE- PI-PURENET. We notice that for WD-TV3D, we apply the same Rademacher matrix to all non-overlapping patches of size 32×32, every measurement is 5-bit uniformly quantized in the range of [-3σ, 3σ], where $\sigma = \frac{0.5}{\sqrt{1024}}$ is the estimated standard deviation of the measurements distribution.

Image quality evaluation: In order to assess the performance of our model compared to different compression methods, we employ the well-known Peak Signal to Noise Ratio (PNSR) besides the Multiscale Structural Similarity (MS-SSIM) [73]. The average PSNR and MS-SSIM performances over the 100 test images are reported in Table IV. It clearly shows that our NQE-PURENET framework delivers better compression quality compared to JPEG and WD-TV3D in both cases patch-based or full-resolution reconstruction, with higher average PSNR/MS-SSIM. The CAE method offers the best compression quality with both these two values much higher than the others –what one would expect–, as it is specifically designed for this task with additional capability to capture both the spatial correlation and the residual information. In particular, compared to the PI-PURENET, the PURENET slightly improves the PSNR but significantly the MS-SSIM, which is easily explainable with respect to the noticeable enhancement in terms of visual quality. This improvement is illustrated in Figure 11 where 4 images in the test dataset are displayed along with the PSNR/MS-SSIM results. We can see the block artifacts in JPEG and PI-PURENET, as they do not take in account the surrounding context of each patch. On the contrary, the PURENET successfully renders the smoothness between patches. This explains why it has MS-SSIM much higher (*i.e.* 0.0172). However, when comparing this method with the CAE and the original image, we clearly observe the lack of finest details, for example the lion's eye and the castle in the third and the fourth columns. This lack can be easily explained as a direct consequence of the quantized nature of the NQE.

TABLE IV: VGA image compression comparison between different methods at the bitrate of 0.25bpp

Metrics / Method	PSNR (dB)	MS-SSIM
JPEG	19.82	0.7610
WD-TV3D	19.67	0.7255
CAE	22.45	0.8959
PI-PURENET (Ours)	20.58	0.7964
PURENET (Ours)	20.76	0.8136

Encoder-Decoder complexity evaluation: While WD-TV3D makes use of standard linear CS at the encoder part (very lightweight in terms of hardware), our NQE consists of convolution, bitshift, nonlinear activation. However they all remain hardware-friendly and perform feed-forward computations within a limited memory budget of 1Mb, with the intensive use of low-precision computations. On the contrary, JPEG involves more complex algorithmic routines since they take advantage of entropy encoding, thus requiring non deterministic computational state machines. On the other hand, CAE uses LSTM cells to deal with various compression rates in a unified model besides an RNN-based entropy coding, so increasing computational needs compared to our approach, in addition to requiring fully floating-point precision. On the decoder side, WD-TV3D uses power-hungry regularization algorithms and CAE decoder remains computationally complex (mainly due to LSTM cells) whereas JPEG decompression is relatively straightforward. In comparison, PURENET is a floating-point convolutional network with dense skip connections but without any recurrent modules. In conclusion, our Full-Resolution PURENET that corresponds to a simple feed-forward DNN, when combined with NQE, achieves a good complexity-quality compromise as it properly recovers images, outperforming JPEG and linear BCS compression.

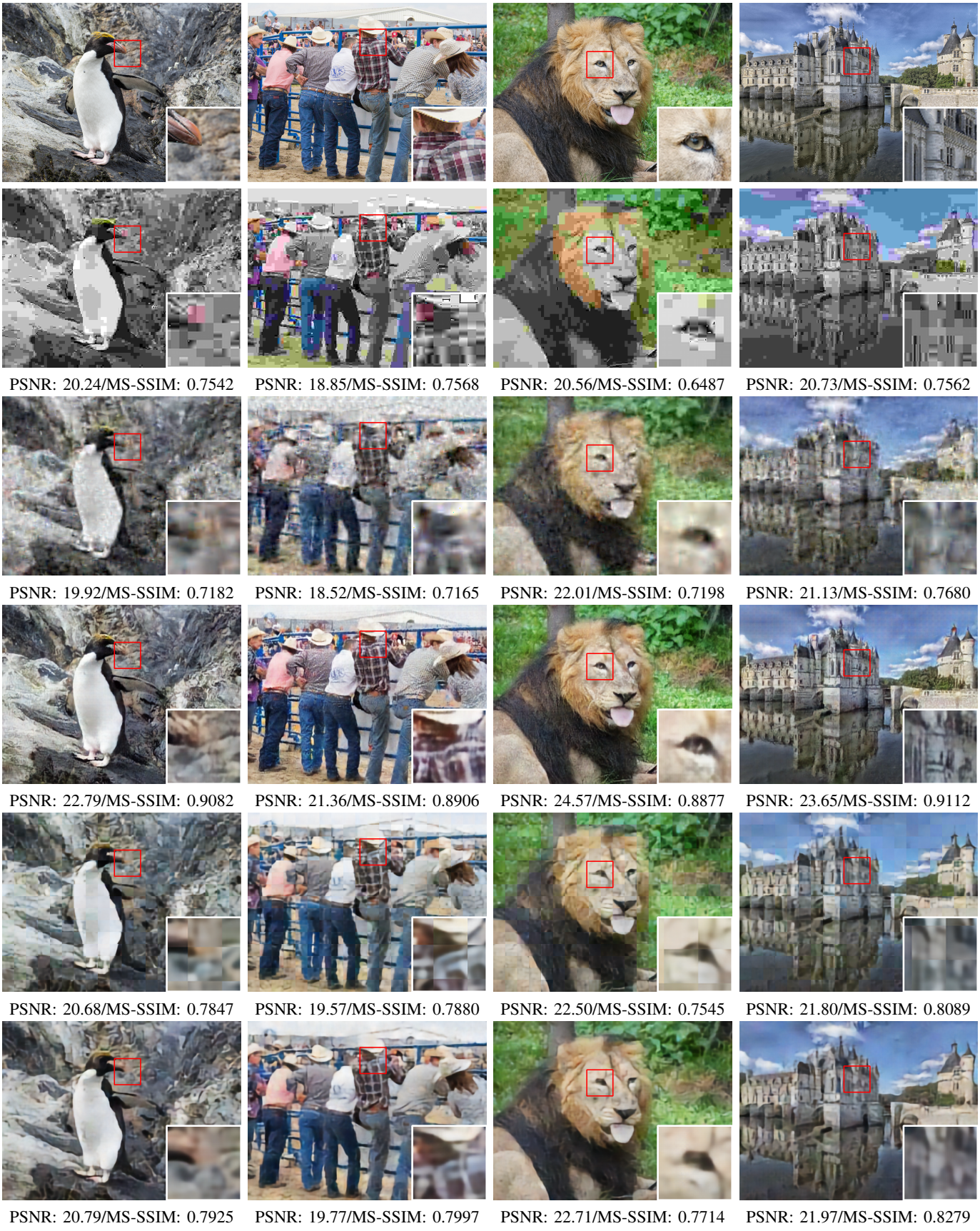

Fig. 11: Image compression results at 0.25bpp on some test images of DIV2K validation dataset. From top to bottom: Original image, JPEG, WD-TV3D, CAE, our NQE- PI-PURENET and NQE- Full-Resolution PURENET.

TABLE V: Effect of the bottleneck and its two alternatives to the overall memory and accuracy.

F	Model memory (Mb) without bottleneck	LFC		RCS + FC		**DWConv + FC (Ours)**	
		Memory (Mb)	Accuracy (%)	Memory (Mb)	Accuracy (%)	Memory (Mb)	Accuracy (%)
32	0.253	0.262	80.53	0.016	77.98	0.018	79.51
64	1.003	1.049	87.69	0.066	86.33	0.070	87.48
128	3.997	4.194	90.75	0.262	89.77	0.270	90.15

VI. Conclusion

In this paper, we presented a mixed-precision CNN topology design approach which is compliant with a future ASIC hardware mapping, enabling to perform both low-complexity image classification and embedded patch-based compression. The reported results therefore demonstrate the possible degree of versatility in terms of application of a specific neural network architecture developed to target an ASIC design. Numerically speaking, our NQE obtains 87.48% accuracy in CIFAR-10 while consuming just over 1Mb of memory whose weights and activations are quantized with a mixed precision approach. In addition, Batch Normalizations are replaced by layer-shared BitShift Normalisations, in order to further ease a possible hardware implementation. In terms of image compression, our PURENET architecture typically deals with patch-based binary coding to perform a collaborative reconstruction, providing images with a relatively high rendering quality at a bitrate of only 0.25bpp. Besides, PSNR and MS-SSIM metrics are better than relevant alternatives such as JPEG and BCS compression schemes. The proposed approach thus exhibits a good quality of service versus its computational complexity, especially for an embedded patch-based image compression. All those results clearly show the advantage of a proper algorithm/hardware co-design to reach the best trade-off between hardware implementation complexity and algorithmic accuracy. Note that a direct extension of this work may involve the use of quantized RNN units to extend NQE compression and classification capabilities to frame sequences, not only still images. Another approach may also be using ensemble learning like [60] (thanks to the reconfigurability) to enhance the compression quality. In other respects and to further improve the performances of the NQE, its topology would also benefit from the use of skip connections so as attention mechanisms.

Appendix A
Effect of the bottleneck alternative

To highlight the efficiency of the proposed alternative to the dense layer, we conduct a study on the impact of different bottleneck types to the model. In details, we compare our proposition with two other settings. The first option is the canonical Fully Connected bottleneck itself denoted as **LFC**, *i.e.* a Learnable FC layer of binary weights ($256F^2$ bits). The second option is another possible alternative solution **RCS+FC**, including the same FC layer like that of LFC, but with fixed weights (*i.e.* not trained) following a zero-mean Rademacher distribution, followed by a small FC layer of binary weights from $4F$ to $4F$ ($16F^2$ bits). This option needs 16 less bits to store the weights compared to LFC, since the Rademacher matrix can be generated on-chip with a pseudo random generator. We evaluate the efficiency of the bottleneck on two metrics: contribution to the total model size, and accuracy. Table V reports our results, clearly demonstrating that a large LFC layer of $256F^2$ consumes more than 50% of the overall memory in all cases. When the model size is small, this parameter-heaviness is crucial to improve the model's performance, explaining why the gap between LFC and its two alternatives is more important at $F = 32$ (more than 1%). On the contrary, its two alternatives contribute only around 6% of the overall memory. The proposed DWConv+FC obtains much higher inference accuracy, while increasing by just a tiny amount the memory budget compared to RCS+FC (< 10kb).

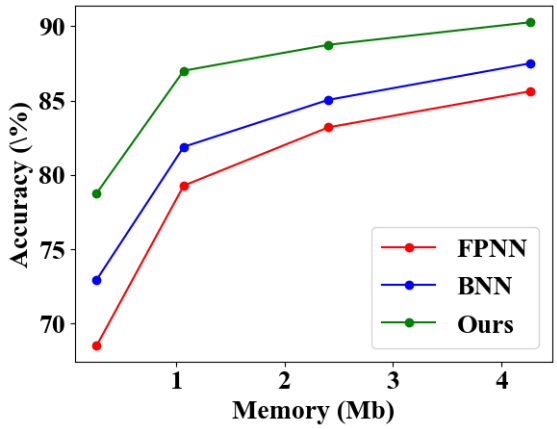

Fig. 12: Memory-accuracy curves of different model's precision (floating-point, binary and ours mixed-precision).

Appendix B
Memory vs Accuracy curves

Apart from the model of 1Mb reported in the V-A, we also provides the accuracy of our framework for three different feature map sizes. We also report comparisons with the original VGG7 model in the case of BNN (respectively FPNN) where all the weights/input activations are binarized (respectively of full-precision). The BN is used for these two configurations, and in the case of BNNs, it has been considered –for a fair comparison– as a Sign Comparison which needs only 6b to store each bias (see Table III). Figure 12 shows that our mixed-precision topology outperforms FPNN and BNN at iso-memory. Our approach exhibits a very large gap, as it provides an efficient design to capture enough discriminant information, compared to the loss due to the binarization of BNN and the "too tiny size" of FPNN. Typically, with a memory budget of 2.4Mb, the average accuracy (3 simulations per point) reaches 89.33% which is much higher than the ones of BNN and FPNN, as well as the reported values in Table III.

REFERENCES

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[2] M. Mathieu, M. Henaff, and Y. LeCun, "Fast Training of Convolutional Networks through FFTs," *International Conference on Learning Representation (ICLR)*, 2014.

[3] J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in *Artificial Neural Networks and Machine Learning – ICANN 2014*, S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, Eds. Cham: Springer International Publishing, 2014, pp. 281–290.

[4] A. Lavin and S. Gray, "Fast Algorithms for Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 4013–4021.

[5] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep Convolutional Neural Network Architecture With Reconfigurable Computation Patterns," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2220–2233, Aug. 2017.

[6] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2016, pp. 367–379, iSSN: 1063-6897.

[7] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, "NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 644–656, Mar. 2019.

[8] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[9] J. Yue, X. Feng, Y. He, Y. Huang, Y. Wang, Z. Yuan, M. Zhan, J. Liu, J.-W. Su, Y.-L. Chung, P.-C. Wu, L.-Y. Hung, M.-F. Chang, N. Sun, X. Li, H. Yang, and Y. Liu, "15.2 A 2.75-to-75.9TOPS/W Computing-in-Memory NN Processor Supporting Set-Associate Block-Wise Zero Skipping and Ping-Pong CIM with Simultaneous Computation and Weight Updating," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 238–240, iSSN: 2376-8606.

[10] T.-H. Kim and J. Shin, "A Resource-Efficient Inference Accelerator for Binary Convolutional Neural Networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 451–455, Jan. 2021, conference Name: IEEE Transactions on Circuits and Systems II: Express Briefs.

[11] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An Ultra-Low Power Convolutional Neural Network Accelerator Based on Binary Weights," in *2016 IEEE Computer Society Annual Symposium on VLSI*, Jul. 2016, pp. 236–241.

[12] J.-H. Kim, C. Kim, K. Kim, and H.-J. Yoo, "An Ultra-Low-Power Analog-Digital Hybrid CNN Face Recognition Processor Integrated with a CIS for Always-on Mobile Devices," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.

[13] A. Verdant, W. Guicquero, N. Royer, G. Moritz, S. Martin, F. Lepin, S. Choisnet, F. Guellec, B. Deschamps, S. Clerc, and J. Chossat, "A 3.0µW@5fps QQVGA Self-Controlled Wake-Up Imager with On-Chip Motion Detection, Auto-Exposure and Object Recognition," in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2, iSSN: 2158-5636.

[14] M. Lefebvre, L. Moreau, R. Dekimpe, and D. Bol, "7.7 A 0.2-to-3.6TOPS/W Programmable Convolutional Imager SoC with In-Sensor Current-Domain Ternary-Weighted MAC Operations for Feature Extraction and Region-of-Interest Detection," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 118–120, iSSN: 2376-8606.

[15] H. Mo, L. Liu, W. Zhu, Q. Li, H. Liu, S. Yin, and S. Wei, "A multi-task hardwired accelerator for face detection and alignment," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 4284–4298, 2020.

[16] P. C. Knag, G. K. Chen, H. E. Sumbul, R. Kumar, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, M. Kar, S. Kim, and R. K. Krishnamurthy, "A 617 TOPS/W All Digital Binary Neural Network Accelerator in 10nm FinFET CMOS," in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2, iSSN: 2158-5636.

[17] X. Chen, J. Xu, and Z. Yu, "A 68-mw 2.2 tops/w low bit width and multiplierless dcnn object detection processor for visually impaired people," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3444–3453, 2019.

[18] D. T. Nguyen, H. Kim, and H.-J. Lee, "Layer-specific optimization for mixed data flow with mixed precision in fpga design for cnn-based object detectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 6, pp. 2450–2464, 2021.

[19] K. Guo, W. Li, K. Zhong, Z. Zhu, S. Zeng, S. Han, Y. Xie, P. Debacker, M. Verhelst, and Y. Wang, "Neural network accelerator comparison," Tech. Rep. [Online]. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/

[20] D. Miyashita, E. G. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *ArXiv*, vol. abs/1603.01025, 2016.

[21] B. Moons, K. Goetschalckx, N. Van Berckelaer, and M. Verhelst, "Minimum energy quantized neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017.

[22] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4107–4115.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv:1603.05279 [cs]*, Aug. 2016.

[24] F. Li, B. Zhang, and B. Liu, "Ternary Weight Networks," *arXiv:1605.04711 [cs]*, Nov. 2016.

[25] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained Ternary Quantization," *arXiv:1612.01064 [cs]*, Feb. 2017.

[26] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura, "Mixed Precision DNNs: All you need is a good parametrization," *arXiv:1905.11452 [cs, stat]*, May 2020.

[27] C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, and M. Welling, "Relaxed Quantization for Discretized NeuralL Networks," p. 15, 2019.

[28] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," p. 16, 2020.

[29] S. K. Esser, J. McKinstry, D. Bablani, R. Appuswamy, and D. Modha, "Learned step size quantization," *ICLR*, 2020.

[30] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Mar. 2015.

[31] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An Always-On 3.8 µ J/86% CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[32] J. Fromm, M. Cowan, M. Philipose, L. Ceze, and S. Patel, "Riptide: Fast end-to-end binarized neural networks," in *Proceedings of Machine Learning and Systems*, 2020.

[33] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019, conference Name: IEEE Journal of Solid-State Circuits.

[34] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, 2020.

[35] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1135–1143.

[36] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017, pp. 1398–1406.

[37] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning Structured Sparsity in Deep Neural Networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2074–2082.

[38] C. Lemaire, A. Achkar, and P.-M. Jodoin, "Structured Pruning of Neural Networks With Budget-Aware Regularization," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 9100–9108.

[39] H.-J. Kang, "Accelerator-aware pruning for convolutional neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2093–2103, 2020.

[40] D. Mittal, S. Bhardwaj, M. M. Khapra, and B. Ravindran, "Recovering from Random Pruning: On the Plasticity of Deep Convolutional Neural Networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2018, pp. 848–857.

[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861 [cs]*, Apr. 2017.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 6848–6856.

[43] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2014.

[44] H. Gao, Z. Wang, and S. Ji, "Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[45] E. Hoffer, I. Hubara, and D. Soudry, "Fix your classifier: the marginal value of training the last weight layer," *ICLR*, 2018.

[46] W. Benjilali, W. Guicquero, L. Jacques, and G. Sicard, "Hardware-Friendly Compressive Imaging Based on Random Modulations Permutations for Image Acquisition and Classification," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 2085–2089, iSSN: 2381-8549.

[47] G. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[48] J. E. Fowler, S. Mun, and E. W. Tramel, 2012.

[49] Y. Oike and A. El Gamal, "CMOS Image Sensor With Per-Column $\sigma\delta$ ADC and Programmable Compressed Sensing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 318–328, Jan. 2013.

[50] Y. Wu, X. Li, Z. Zhang, X. Jin, and Z. Chen, "Learned block-based hybrid image compression," *ArXiv*, vol. abs/2012.09550, 2020.

[51] A. Beck and M. Teboulle, "Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.

[52] W. Guicquero, A. Verdant, and A. Dupret, "High-order incremental sigma–delta for compressive sensing and its application to image sensors," *Electronics Letters*, vol. 51, no. 19, pp. 1492–1494, 2015.

[53] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "ReconNet: Non-Iterative Reconstruction of Images from Compressively Sensed Measurements," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[54] H. Yao, F. Dai, D. Zhang, Y. Ma, S. Zhang, and Y. Zhang, "Dr^2-net: Deep residual reconstruction network for image compressive sensing," *CoRR*, vol. abs/1702.05743, 2017. [Online]. Available: http://arxiv.org/abs/1702.05743

[55] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. C. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural network," in *ICLR*, 2016.

[56] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full Resolution Image Compression with Recurrent Neural Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jul. 2017.

[57] M. H. Baig, V. Koltun, and L. Torresani, "Learning to inpaint for image compression," in *NIPS*, 2017.

[58] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3214–3223.

[59] C. Cai, L. Chen, X. Zhang, and Z. Gao, "Efficient variable rate image compression with multi-scale decomposition network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 12, pp. 3687–3700, 2019.

[60] Y. Wang, D. Liu, S. Ma, F. Wu, and W. Gao, "Ensemble learning-based rate-distortion optimization for end-to-end image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 3, pp. 1193–1207, 2021.

[61] Y. Ma, H. Xiong, Z. Hu, and L. Ma, "Efficient super resolution using binarized neural network," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 694–703, 2019.

[62] J. Xin, N. Wang, X. Jiang, J. Li, H. Huang, and X. Gao, "Binarized Neural Network for Single Image Super Resolution," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020.

[63] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," p. 60, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[64] Y. Cai, T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Low bit-width convolutional neural network on rram," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 7, pp. 1414–1427, 2020.

[65] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[67] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[68] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv:1308.3432 [cs]*, Aug. 2013.

[69] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Artificial Intelligence and Statistics*. Proceedings of Machine Learning Research (PMLR), Feb. 2015.

[70] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.

[71] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[72] Y. Xiao, J. Yang, X. Yuan, ,Institute of Applied Mathematics, Henan University, Kaifeng 475004, ,Department of Mathematics, Nanjing University, Nanjing, 210093, and ,Department of Mathematics, Hong Kong Baptist University, Hong Kong, "Alternating algorithms for total variation image reconstructionfrom random projections," *Inverse Problems & Imaging*, vol. 6, no. 3, pp. 547–563, 2012.

[73] Z. Wang, E. P. Simoncelli, and A. Bovik, "Multi-scale structural similarity for image quality assessment," in *In Signals, Systems and Computers 2004. Conference Record of the Thirty-Seventh Asilomar Conference*, 2003.