Feature Map Transform Coding for Energy-Efficient CNN Inference

Brian Chmiel*†, Chaim Baskin†, Evgenii Zheltonozhskii†, Ron Banner*, Yevgeny Yermolin†,
Alex Karbachevsky†, Alex M. Bronstein† and Avi Mendelson†
*Intel – Artificial Intelligence Products Group (AIPG), Haifa, Israel
†Technion – Israel Institute of Technology, Haifa, Israel
{brian.chmiel, ron.banner}@intel.com
{chaimbaskin, alex.k, bron, avi.mendelson}@cs.technion.ac.il
{evgeniizh, yevgeny_ye}@campus.technion.ac.il

***Abstract*—Convolutional neural networks (CNNs) achieve state-of-the-art accuracy in a variety of tasks in computer vision and beyond. One of the major obstacles hindering the ubiquitous use of CNNs for inference on low-power edge devices is their high computational complexity and memory bandwidth requirements. The latter often dominates the energy footprint on modern hardware. In this paper, we introduce a lossy transform coding approach, inspired by image and video compression, designed to reduce the memory bandwidth due to the storage of intermediate activation calculation results. Our method does not require fine-tuning the network weights and halves the data transfer volumes to the main memory by compressing feature maps, which are highly correlated, with variable length coding. Our method outperform previous approach in term of the number of bits per value with minor accuracy degradation on ResNet-34 and MobileNetV2. We analyze the performance of our approach on a variety of CNN architectures and demonstrate that FPGA implementation of ResNet-18 with our approach results in a reduction of around 40% in the memory energy footprint, compared to quantized network, with negligible impact on accuracy. When allowing accuracy degradation of up to 2%, the reduction of 60% is achieved. A reference implementation accompanies the paper.**

***Index Terms*—Neural networks, quantization, FPGA, energy-efficient inference, convolutional neural networks**

I. Introduction

Deep neural networks have established themselves as the first-choice tool for a wide range of applications. Neural networks have shown phenomenal results in a variety of tasks in a broad range of fields such as computer vision, computational imaging, and image and language processing. Despite deep neural models impressive performance, the computation and computational requirements are substantial for both training and inference phases. So far, this fact has been a major obstacle for the deployment of deep neural models in applications constrained by memory, computational, and energy resources, such as those running on embedded systems.

To alleviate the energy cost, custom hardware for neural network inference, including FPGAs and ASICs, is actively being developed in recent years. In addition to providing better energy efficiency per arithmetic operation, custom hardware offers more flexibility in various strategies to reduce the computational and storage complexity of the model inference, for example by means of quantization [3, 18, 20] and pruning [16, 23, 37]. In particular, quantization to very low precision is especially efficient on custom hardware where arbitrary precision arithmetic operations require proportional resources. To prevent accuracy degradation, many approaches have employed training the model with quantization constraints or modifying the network structure.

A recent study [43] has shown that almost 70% of the energy footprint on such hardware is due to data movement to and from the off-chip memory. Amounts of data typically need to be transferred to and from the RAM and back during the forward pass through each layer, since the local memory is too small to store all the feature maps. By reducing the number of bits representing these data, existing quantization techniques reduce the memory bandwidth considerably. However, to the best of our knowledge, none of these methods exploit the high amount of interdependence between the feature maps and spatial locations of the compute activations.

Contributions. In this paper, we propose a novel scheme based on transform-domain quantization of the neural network activations followed by lossless variable length coding. The method does not require neither backpropogation nor training data except for a one calibration batch. We demonstrate that this approach reduces memory bandwidth by 40% when applied in the post-training regime (i.e., without fine-tuning) with small computational overhead and no accuracy degradation. Relaxing the accuracy requirements increases bandwidth savings to 60%. Moreover, we outperform previous methods in term of number bit per value with minor accuracy degradation. A detailed evaluation of various ingredients and parameters of the proposed method is presented. We also demonstrate a reference hardware implementation that confirms a reduction in memory energy consumption during inference.

II. Related work

a) Quantization: Low-precision representation of the weights and activations is a common means of reducing com-

The research was funded by ERC StG RAPID and Hiroshi Fujiwara Technion Cyber Security Research Center.

The first three authors contribute equally to this work.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

putational complexity. On appropriate hardware, this typically results in the reduction of the energy footprint as well. It has been demonstrated that in standard architectures quantization down to 16 [15] or 8 bits [20, 22, 42] per parameter is practically harmless. However, further reduction of bitwidth requires non-trivial techniques [26, 44], often with adverse impact on training complexity. Lately, the quantization of weights and activations of neural networks to 2 bits or even 1 [18, 30] has attracted the attention of many researchers. While the performance of binary (i.e., 1-bit) neural networks still lags behind their full-precision counterparts [10, 29], existing quantization methods allow 2-4 bit quantization with a negligible impact on accuracy [6, 7, 11].

Quantizing the neural network typically requires introducing the quantizer model at the training stage. However, in many applications the network is already trained in full precision, and there is no access to the training set to configure the quantizer. In such a *post-training* regime, most quantization methods employ *statistical clamping*, i.e., the choice of quantization bounds based on the statistics acquired in a small test set. Migacz [25] proposed using a small calibration set to gather activation statistics and then randomly searching for a quantized distribution that minimizes the Kullback-Leibler divergence to the continuous one. Gong et al. [12], on the other hand, used the L_∞ norm of the tensor as a threshold. Lee et al. [22] employed channel-wise quantization and constructed a dataset of parametric probability densities with their respective quantized versions; a simple classifier was trained to select the best fitting density. Banner et al. [2] derived an approximation of the optimal threshold under the assumption of Laplacian or Gaussian distribution of the weights, which achieved single-percentage accuracy reduction for 8-bit weights and 4-bit activations. Meller et al. [24] showed that the equalization of channels and removal of outliers improved quantization quality. Choukroun et al. [8] used one-dimensional line-search to evaluate an optimal quantization threshold, demonstrating state-of-the-art results for 4-bit weight and activation quantization.

b) Influence of memory access on energy consumption: Yang et al. [43] studied the breakdown of energy consumption in CNN inference. For example, in GoogLeNet [34] arithmetic operations consume only 10% of the total energy, while feature map transfers to and from an external RAM amount to about 68% of the energy footprint. However, due to the complexity of real memory systems, not every method that decreases the sheer memory bandwidth will necessarily yield significant improvement in power consumption. In particular, it depends on computational part optimization: while memory performance is mainly defined by the external memory chip, better optimization of computations will lead to higher relative energy consumption of the memory. For example, while Ansari and Ogunfunmi [1] reported a 70% bandwidth reduction, the dynamic power consumption decreased by a mere 2%.

Xiao et al. [40] proposed fusing convolutional layers to reduce the transfer of feature maps. In an extreme case, all layers are fused into a single group. A similar approach was adopted by Xing et al. [41], who demonstrated a hardware

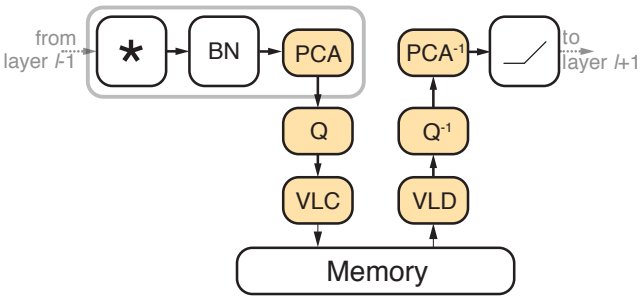


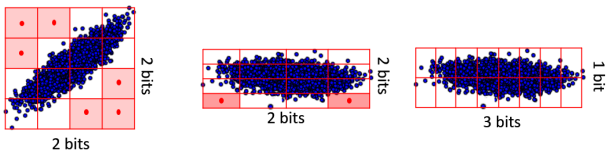
Fig. 1: High-level flow diagram of the encoder-decoder chain. PCA and BN are folded into the convolution weights (denoted by $*$), resulting in a single convolution (boxed in grey).

design that does not write any intermediate results into the off-chip memory. This approach achieved approximately 15% runtime improvement for ResNet and state-of-the-art throughput. However, the authors did not compare the energy footprint of the design with the baseline. Morcel et al. [27] demonstrated that using on-chip cache cuts down the memory bandwidth and thus reduces power consumption by an order of magnitude. In addition, Jouppi et al. [21] noted that not only the power consumption but also the speed of DNN accelerators is memory- rather than compute-bound. This is confirmed by Wang et al. [38], who also demonstrated that increasing computation throughput without increasing memory bandwidths barely affects latency.

c) Network compression: Lossless coding and, in particular, variable length coding (VLC), is a way to reduce the memory footprint without compromising performance. In particular, Han et al. [16] proposed using Huffman coding to compress network weights, alongside quantization and pruning. Wijayanto et al. [39] proposed using the more computationally-demanding algorithm DEFLATE (LZ77 + Huffman) to further improve compression rates. Chandra [5] used Huffman coding to compress feature maps and thus reduce memory bandwidth. Gudovskiy et al. [14] proposed passing only the lower-dimensional feature maps to the memory to reduce the bandwidth. Cavigelli and Benini [4] proposed using RLE-based algorithm to compress sparse network activations.

III. Transform-Domain Compression

In what follows, we briefly review the basics of lossy transform coding. For a detailed overview of the subject, we refer the reader to Goyal [13]. Let $\mathbf{x} = (x_1, \ldots, x_n)$ represent the values of the activations of a NN layer in a block of size $n = W \times H \times C$ spanning, respectively, the horizontal and the vertical dimensions and the feature channels. Prior to being sent to memory, the activations, $\mathbf{x}$, are encoded by first undergoing an affine transform, $\mathbf{y} = \mathcal{T}\mathbf{x} = \mathbf{T}(\mathbf{x} - \boldsymbol{\mu})$; the transform coefficients are quantized by a scalar quantizer, $\mathcal{Q}_\Delta$, whose strength is controlled by the step size Δ, and subsequently coded by a lossless variable length coder (VLC). We refer to the length in bits of the resulting code, normalized by n as to the *average rate*, R_Δ. To decode the activation vector, a



(a) (b) (c)

Fig. 2: Vector quantization in 2D case. **(a)** A pair of correlated channels on a scatter plot. All values in a cell are mapped to the center of the cell; hence, small cells induce small quantization noise. Several bins are empty (red cells); **(b)** Decorrelation improves utilization since the cells are smaller now; **(c)** Forcing equal bin size along all dimensions further improves utilization. Instead of restricting both channel dynamic range to be divided into same number of bins, we use uniform bin size along all dimensions. VLC allows to further compress the representation since the channels with smaller dynamic range have are mapped mostly to a few most probable bins.

variable length decoder (VLD) is applied first, followed by the inverse quantizer and the inverse transform. The resulting decoded activation, $\hat{\mathbf{x}} = \mathcal{T}^{-1}\mathcal{Q}_\Delta^{-1}(\mathcal{Q}_\Delta(\mathcal{T}\mathbf{x}))$, typically differs from $\mathbf{x}$; the discrepancy is quantified by a *distortion*, D_Δ. The functional relation between the rate and the distortion is controlled by the quantization strength, Δ, and is called *rate-distortion curve*.

Classical rate-distortion analysis in information theory assumes the MSE distortion, $D = \frac{1}{n}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$. While in our case the measure of distortion is the impact on the task-specific loss, we adopt the Euclidean distortion for two reasons: firstly, it is well-studied and leads to simple expressions for the quantizer; and, secondly, computing loss requires access to the training data, which are unavailable in the post-training regime.

A crucial observation justifying transform coding is the fact that significant statistical dependence usually exists between the x_i [9]. We model this fact by asserting that the activations are jointly Gaussian, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with the covariance matrix $\boldsymbol{\Sigma}$, whose diagonal elements are denoted by σ_i^2. Statistical dependence corresponds to non-zero off-diagonal elements in $\boldsymbol{\Sigma}$. The affinely transformed $\mathbf{y} = \mathbf{T}(\mathbf{x} - \boldsymbol{\mu})$ is also Gaussian with the covariance matrix $\boldsymbol{\Sigma}' = \mathbf{T}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{T}$. The distortion is minimized over orthonormal matrices by $\mathbf{T} = \boldsymbol{\Sigma}^{-1/2}$ diagonalizing the covariance [13]. The latter is usually referred to as the Karhunen-Loeve transform (KLT) or principal component analysis (PCA). The corresponding minimum distortion is $D^*(R) = \frac{\pi e}{6}\det(\boldsymbol{\Sigma})^{1/n}2^{-2R}$. Since the covariance matrix is symmetric, $\mathbf{T}$ is orthonormal, implying $\mathbf{T}^{-1} = \mathbf{T}^{\mathrm{T}}$.

In Fig. 2, a visualization of 2D vector quantization is shown. For correlated channels (Fig. 2a), many 2D quantization bins are not used since they contain no values. Linear transformation (Fig. 2b) provides improved quantization error for correlated channels by getting rid of those empty bins.

A. Implementation

In what follows, we describe an implementation of the transform coding scheme at the level of individual CNN layers. The convolutional layer depicted in Fig. 1 comprises a bank of convolutions (denoted by $*$ in the Figure) followed by batch normalization (BN) that is computed on an incoming input stream. The output of BN is a 3D tensor that is subdivided into 3D blocks to which the transform coding is applied. Each such block is sent to an encoder, where it undergoes PCA, scalar quantization and VLC. The bit stream at the VLC output has a lower rate than the raw input and is accumulated in the external memory. Once all the output of the layer has been stored in the memory, it can be streamed as the input to the following layer. For that purpose, the inverse process is performed by the decoder: a VLD produces the quantized levels that are scaled back to the transform domain, and an inverse PCA is applied to reconstruct each of the activation blocks. The layer non-linearity is then applied, and the activations are used as an input to the following layer. While the location of the nonlinearity could also precede the encoder, our experiments show that the described scheme performs better.

a) Linear transform: We have explored different sizes of blocks for the PCA transform and found $1\times1\times C$ to be optimal (the ablation study is shown in Section IV-B). Moreover, this choice allows to optimize the calculation of the transformation: applying same linear transformation to every $1 \times 1 \times C$ block is a convolution with 1×1 kernel, which can be calculated very efficiently. This allows further optimization: as depicted in Fig. 1, the convolution bank of the layer, BN and PCA can be folded [20] into a single operation, offering also an advantage in the arithmetic complexity.

The PCA matrix is pre-computed, as its computation requires the expensive eigendecomposition of the covariance matrix. The covariance matrix is estimated on a small batch of (unlabeled) training or test data and can be updated online. Estimation of the covariance matrix for all layers at once is problematic since quantizing activations in the l-th layer alters the input to the $l+1$-st layer, resulting in a biased estimation of the covariance matrix in the $l+1$-st layer. To avoid this, we calculate the covariance matrix layer by layer, gradually applying the quantization: at iteration i, first $i-1$ layers perform PCA transformation and only i layer covariance matrix estimation is updated. The PCA matrix is calculated after quantization of the weights is performed, and is itself quantized to 8 bits.

b) Quantization: For transformed feature maps we use a uniform quantization, where the dynamic range is determined according to the channel with the highest variance. Since all channels have an equal quantization step, entropy of the low-variance channels is significantly reduced.

c) Variable length coding: The theoretical rate associated with a discrete random variable, Y (the output of the quantizer), is given by its entropy $H(X) = -\mathbb{E}\log_2 X = -\sum_i p(x_i)\log_2 p(x_i)$. This quantity constitutes the lower bound on the amount of information required for lossless compression of Y. We use Huffman codes, which are a practical variable length coding method [36], achieving the rates bounded by $H(X) \le R \le H(X)+1$ (see Fig. 3 for a comparison of the theoretical rates to the ones attained by Huffman codes).

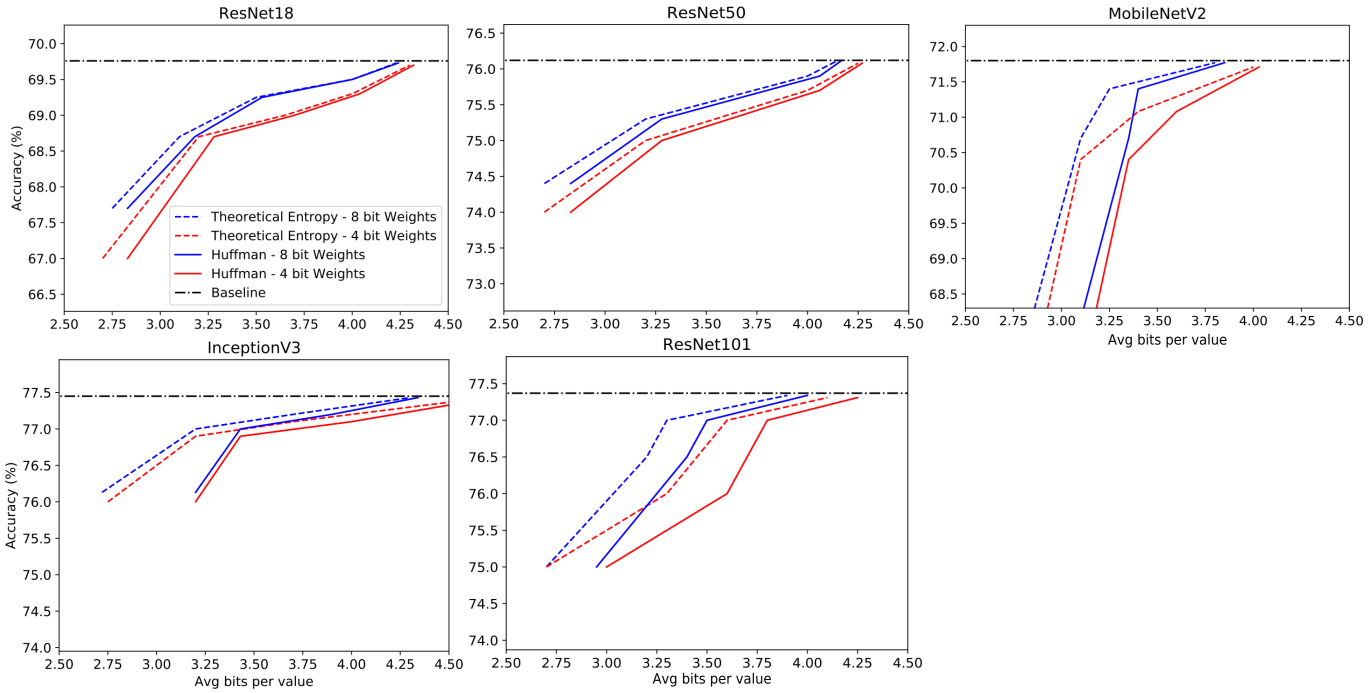


Fig. 3: Rate-distortion curves for ResNet-18, ResNet-50, ResNet-101, Inception V3 and MobileNetV2 architectures with 8-(blue) and 4-bit (red) weight quantization. Distortion is evaluated in terms of top-1 accuracy on ImageNet. Dashed lines represent rates obtained by Huffman VLC, while solid lines represent theoretical rates (entropy).

TABLE I: Comparison with EBPC [4]. While EBPC does not affect performance of the network, our method allows better compression by exploiting rate-distortion tradeoff.

Architecture	Method	Activations (avg. number of bits per value)	Accuracy
ResNet-34	EBPC	3.33	73.3%
	Our method	3.9	72.9%
	Our method	**3.11**	72.1%
MobileNetV2	EBPC	3.64	71.7%
	Our method	3.8	71.6%
	Our method	**3.25**	71.4%

d) 1×1 *and grouped convolutions:* While for regular 3×3 convolutions the computational overhead is small, there are two useful cases in which this is not true: 1×1 and grouped convolutions. For 1×1 convolutions the overhead is higher: the transformation requires as much computation as the convolution itself. Nevertheless, it can still be feasible in the case of energy-efficient computations. In the case of grouped convolutions, it is impossible to fold the transformation inside the convolution. However, in the common case when the grouped convolution is followed by a regular one, we can change the order of operations: we perform BN, activation and transformation before writing to the memory. This way, the inverse transformation can be folded inside the following convolution.

IV. Experimental Results

We evaluate the proposed framework on common CNN architectures that have achieved high performance on the ImageNet benchmark. The inference contains 2 stages: a calibration stage, on which the linear transformation is learned based on a single batch of data, and the test stage.

a) Full model performance: We evaluted our method on different CNN architectures: ResNet-18, 50, 101 [17]; MobileNetV2 [32]; and InceptionV3 [35]. Specifically, MobileNetV2 is known to be unfriendly to activation quantization [33]. Performance was evaluated on ImageNet dataset [31] on which the networks were pre-trained. The proposed method was applied to the outputs of all convolutional layers, while the weights were quantized to either 4 or 8 bits (two distinct configurations) using the method proposed by Banner et al. [2]. Rates are reported both in terms of the entropy value and the average length of the feature maps compressed using Huffman VLC in Fig. 3. We observed that higher compression is achieved for covariance matrices with fast decaying eigenvalues describing low-dimensional data. A full analysis can be found in Section IV-A.

b) Comparison to other methods: We compare the proposed method with other post-training quantization methods: ACIQ [2], GEMMLOWP [19], and KLD [25]. Note that our method can be applied on top of any of them to further reduce the memory bandwidth. For each method, we varied the bitwidth and chose the smallest one that attained top-1 accuracy within 0.1% from the baseline and measured the entropy of the activations. Our method reduces, in average, 36% of the memory bandwidth relatively to the best competing methods; the full comparison can be found in Table II.

As for other memory bandwidth reduction methods, our

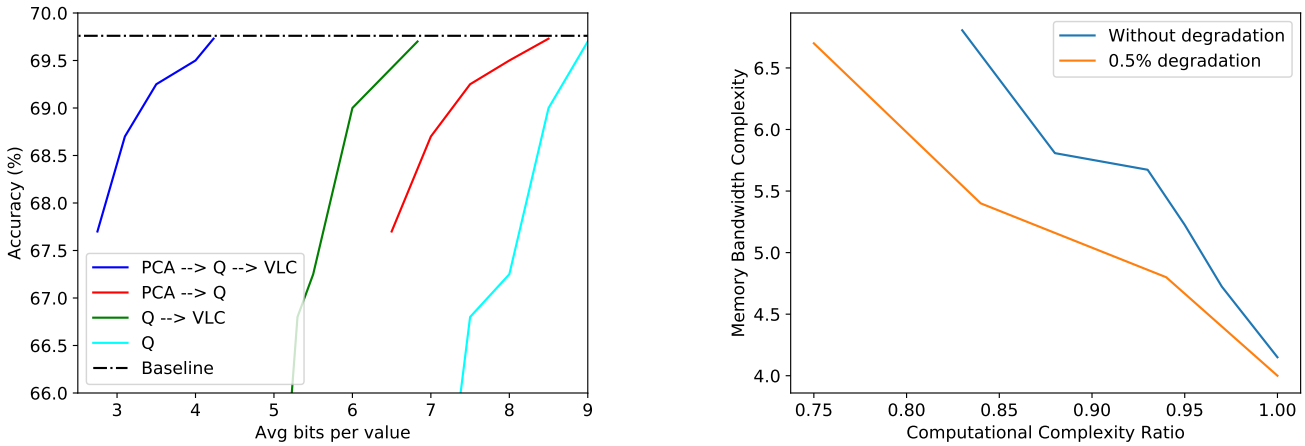


Fig. 4: Ablation study of the proposed encoder on ResNet-18. Left: rate-distortion curve with different encoder configurations. Theoretical rates are reported; top-1 accuracy is used as the distortion measure. Right: theoretical memory rate in bits per value achieved for different levels of PCA truncation for baseline and 0.5% lower than baseline top-1 accuracy.

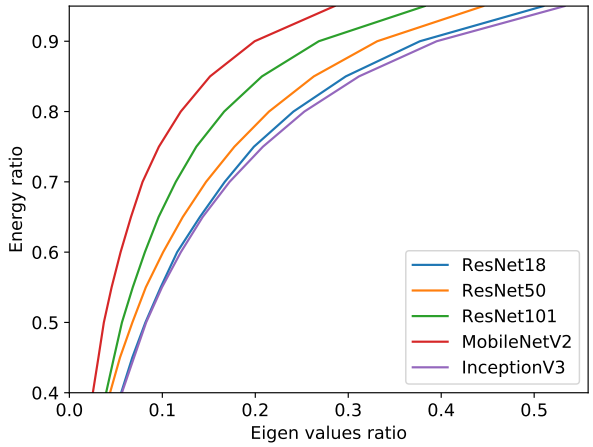


Fig. 5: Analysis of the eigenvalues ratio that are needed to achieve energy ratio, means cumulative sum of the eigenvalues.

method shows better performance than Cavigelli and Benini [4] at the expense of performance degradation (Table I). The performance difference is smaller in MobileNetV2, since mobile architectures tend to be less sparse [28], making RLE less efficient. While the method proposed by Gudovskiy et al. [14] requires fine-tuning, our method, although introducing computational overhead, can be applied to any network without such limitations. In addition, similarly to [14] it is possible to compress only part of the layers in which the activation size is most significant.

c) Ablation study: An ablation study was performed using ResNet-18 to study the effect of different ingredients of the proposed encoder-decoder chain. The following settings were compared:

- only quantization of the feature maps with standard uniform quantization (Q);
- applying PCA transformation to the feature maps and quantizing the latter (PCA→Q);
- applying quantization to the feature maps and then compressing them using VLC, without PCA (Q→VLC);
- the full suggested method (PCA→Q→VLC).

The resulting rate-distortion curves are compared in Fig. 4

TABLE II: Comparison of our method against three known post-training quantization methods ((i) ACIQ [2]; (ii) GEMMLOWP [19]; (iii) KLD [25]. We report the smallest bit per value for which degradation is at most 0.1% of the baseline.

Architecture	Weights (bits)	Method	Activations (avg number of bits per value)
ResNet-50	8	GEMMLOWP	6.88
		ACIQ	6
		KLD	6.3
		Our method	**4.15**
	4	GEMMLOWP	6.93
		ACIQ	6.2
		KLD	6.52
		Our method	**4.25**
Inception V3	8	GEMMLOWP	6.93
		ACIQ	6.2
		KLD	6.43
		Our method	**4.3**
	4	GEMMLOWP	6.97
		ACIQ	6.3
		KLD	6.35
		Our method	**4.6**
MobileNetV2	8	GEMMLOWP	8.6
		ACIQ	7.5
		KLD	7.8
		Our method	**3.8**
	4	GEMMLOWP	8.8
		ACIQ	7.85
		KLD	8.1
		Our method	**4**

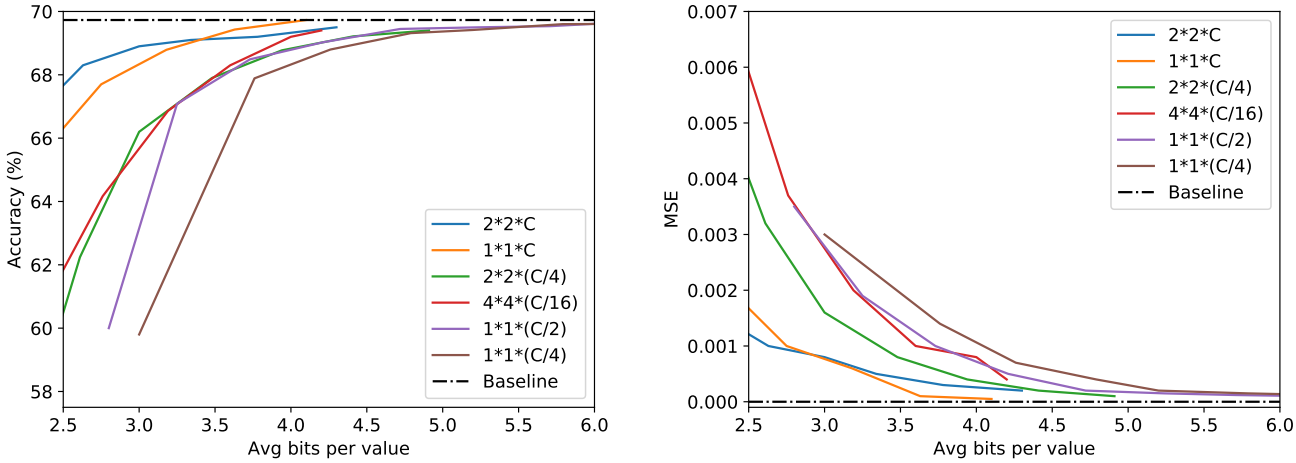


Fig. 6: The influence of the block shape on the top-1 validation accuracy (left) and MSE (right) of ResNet-18 on ImageNet. Our experiments show that for the same size, the most efficient shape is $1 \times 1 \times C$, taking advantage of the correlations across different feature maps at the same spatial location. Theoretical rates are shown.

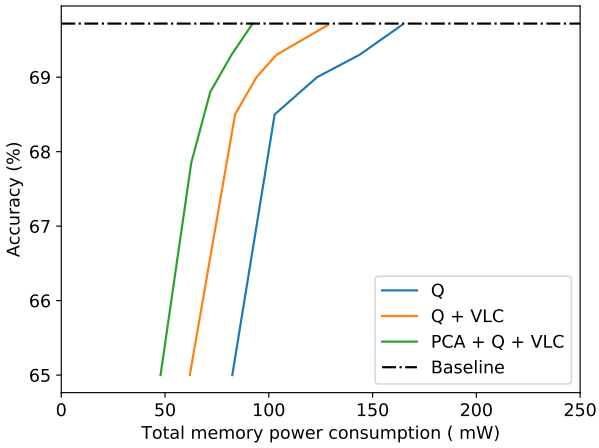


Fig. 7: Top-1 accuracy on ResNet18 ImageNet vs. power consumption of our hardware implementation. Each point represents a different quantization rate.

(left). Our results confirm the previous result of Chandra [5], suggesting that VLC applied to quantized activations can significantly reduce memory bandwidth. They further show that a combination with PCA makes the improvement dramatically bigger. In addition, we analyze the effect of truncating the least significant principal components, which reduces the computational overhead of PCA. Fig. 4 (right) shows the tradeoff between the computational and memory complexities, with baseline accuracy and 0.5% below the baseline.

In Table II we show the comparison of our method with other post-training quantization method. For fair comparison, we add to all compared methods a VLC and show the minimum amount of information that is required to be transferred to the memory.

A. *Eigenvalues analysis*

The eigenvalues of the covariance matrix is a measure of the dispersal of the data. If high energy ratio, means the cumulative sum of eigenvalues divided by the total sum, can be expressed with small part of the eigenvalues, the data is less dispersal and therefore more compressible. In figure 5 we analyze the covariance energy average ratio in all layers of different architectures. The interesting conclusion is that the ability of compression with the suggested algorithm is correlated with the covariance energy average ratio , means that for new architectures we can look only at the energy ratio of the activation to measure our ability of compression.

B. *Block shape and size*

Fig. 6 shows the rate-distortion curves for blocks of the same size allocated differently to each of the three dimensions; the distortion is evaluated both in terms of the MSE and the network classification accuracy. The figure demonstrates that optimal performance for high accuracy is achieved with $1 \times 1 \times C = n$ blocks, suggesting that the correlation between the feature maps is higher than that between spatially adjacent activations. For lower accuracy, bigger blocks are even more efficient, but the overhead of 4 times bigger block is too high. Experiments reported later in the paper set the block size to values between 64 to 512 samples.

V. Hardware Implementation

In order to verify the practical impact of the proposed approach of reducing feature map entropy to save total energy consumption, we implemented the basic building blocks of a pre-trained ResNet-18, with weights and activation quantized to 8 bit, on Intel Stratix-10 FPGA, part number 1SX280LU3F50I2VG. From logic utilization and memory energy consumption (exact numbers are shown in Table III) of convolutional layers we conclude that our method add minor

TABLE III: Logic utilization and memory energy consumption of layers of various widths on Intel's Stratix10 FPGA. Clock frequency was fixed at 160MHz for each design. In LUTs and DSP we present the % of total resources. In Power and Bandwidth we present the total number (% saving comparison to regular quantization)

# channels	Method	LUTs	DSPs	Energy (μJ)	Bandwidth (Gbps)
64	Quantization	19K	960	225.93	1.28
	Q+VLC	19K	960	173.44 (-23%)	0.96 (-25%)
	Q+VLC+PCA	19.5K (+5%)	1056(+10%)	100.6 (-44%)	0.68 (-46.8%)
128	Quantization	43K	2240	112.96	1.28
	Q+VLC	43K	2240	148 (-17%)	1.04 (-18.7%)
	Q+VLC+PCA	45K(+4.3%)	2366(+5.6%)	117 (-35%)	0.83 (-35.1%)
256	Quantization	91K	4800	56.5	1.28
	Q+VLC	91K	4800	46.8 (-17.2%)	1.03 (-19.5%)
	Q+VLC+PCA	93K(+4%)	5059(+5.4%)	103 (-42.8%)	0.73 (-42.9%)
512	Quantization	182K	9600	28.2	1.28
	Q+VLC	182K	9600	23.9 (-15.6%)	1.05 (-17.9%)
	Q+VLC+PCA	186K(+2%)	10051(+4.7%)	91 (-49.5%)	0.66 (-48.4%)

computational overhead in contrast to significant reduction in memory energy consumption. Fig. 7 shows total energy consumption for a single inference of ResNet-18 on ImageNet. In particular, our method is more efficient for higher accuracies, where the redundancy of features is inevitably higher. We further noticed that our approach reached real-time computational inference speed (over 40 fps). The reduction in memory bandwidth can be exploited by using cheaper, slower memory operating at lower clock speeds, which may further reduce its energy footprint. Source files for hardware implementation can found at reference implementation.

We have implemented ResNet-18 using a Stratix-10 FPGA by Intel, part number 1SX280LU3F50I2VG. The memory used for energy calculation is the Micron 4Gb x16 - MT41J256M16. Current consumption of the DDR was taken from the data sheet for read and write operation, and was used to calculate the energy required to transfer the feature maps in each layer. The script for calculating the energy consumption accompanies reference implementation.

In our design each convolutional layer of ResNet-18 is implemented separately. In each layer we calculate 1 pixel of 1 output feature each clock. For example, the second layer has 64 input and 64 output feature maps, thus it takes $64 \times (56 \times 56)$ clock cycles to calculate the output before moving to the next layer.

We read the input features only once by caching the pixels and reusing them from internal memory, and only reloading the weights of the filters in the current layer.

VI. Conclusion

This paper presents a proof-of-concept of energy optimization in NN inference hardware by lossy compression of activations prior to their offloading to the external memory. Our method uses transform-domain coding, exploiting the correlations between the activation values to improve their compressibility, reducing bandwidth by approximately 25% relative to VLC and approximately 40% relative to an 8-bit baseline without accuracy degradation and by 60% relative to an 8-bit baseline with less than 2% accuracy degradation. The computational overhead required for additional linear transformation is relatively small and the proposed method can be easily applied on top of any existing quantization method.

References

[1] A. Ansari and T. Ogunfunmi, "Selective data transfer from drams for cnns," in *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2018, pp. 1–6.

[2] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," 2018.

[3] C. Baskin, E. Schwartz, E. Zheltonozhskii, N. Liss, R. Giryes, A. M. Bronstein, and A. Mendelson, "Uniq: Uniform noise injection for the quantization of neural networks," *arXiv preprint arXiv:1804.10969*, 2018.

[4] L. Cavigelli and L. Benini, "Extended bit-plane compression for convolutional neural network accelerators," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, March 2019, pp. 279–283.

[5] M. Chandra, "Data bandwidth reduction in deep neural network socs using history buffer and huffman coding," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*. IEEE, 2018, pp. 1–3.

[6] J. Choi, P. I.-J. Chuang, Z. Wang, S. Venkataramani, V. Srinivasan, and K. Gopalakrishnan, "Bridging the accuracy gap for 2-bit quantized neural networks (qnn)," *arXiv preprint arXiv:1807.06964*, 2018. [Online]. Available: https://arxiv.org/abs/1807.06964

[7] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[8] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," 2019.

[9] M. Cogswell, F. Ahmed, R. B. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. [Online]. Available: http://arxiv.org/abs/1511.06068

[10] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," *arXiv preprint arXiv:1904.02823*, 2019. [Online]. Available: https://arxiv.org/abs/1904.02823

[11] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," *arXiv preprint arXiv:1905.03696*, 2019.

[12] J. Gong, H. Shen, G. Zhang, X. Liu, S. Li, G. Jin, N. Maheshwari, E. Fomenko, and E. Segal, "Highly efficient 8-bit low precision inference of convolutional neural networks with intelcaffe," *Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning - ReQuEST '18*, 2018. [Online]. Available: http://dx.doi.org/10.1145/3229762.3229763

[13] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, Sep. 2001.

[14] D. Gudovskiy, A. Hodgkinson, and L. Rigazio, "Dnn feature map compression using learned representation over gf(2)," in *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[15] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1737–1746. [Online]. Available: http://proceedings.mlr.press/v37/gupta15.html

[16] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, 2018.

[19] B. Jacob and P. Warden, "gemmlowp: a small self-contained low-precision gemm library," 2017.

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html

[21] N. P. Jouppi, C. Young, N. Patil, D. A. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, R. C. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 1–12.

[22] J. H. Lee, S. Ha, S. Choi, W.-J. Lee, and S. Lee, "Quantization for rapid deployment of deep neural networks," 2018.

[23] C. Louizos, M. Welling, and D. P. Kingmao, "Learning sparse neural networks through l0 regularization," *ICLR*, 2018.

[24] E. Meller, A. Finkelstein, U. Almog, and M. Grobman, "Same, same but different-recovering neural network quantization error through weight factorization," *arXiv preprint arXiv:1902.01917*, 2019.

[25] S. Migacz, "8-bit inference with tensorrt," 2017. [Online]. Available: http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf

[26] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1ZvaaeAZ

[27] R. Morcel, H. Hajj, M. A. R. Saghir, H. Akkary, H. Artail, R. Khanna, and A. Keshavamurthy, "Feathernet: An accelerated convolutional neural network design for resource-constrained fpgas," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 2, pp. 6:1–6:27, Mar. 2019. [Online]. Available: http://doi.acm.org/10.1145/3306202

[28] M. S. Park, X. Xu, and C. Brick, "Squantizer: Simultaneous learning for both sparse and low-precision neural networks," *arXiv preprint arXiv:1812.08301*, 2018.

[29] H. Peng and S. Chen, "Bdnn: Binary convolution neural networks for fast object detection," *Pattern Recognition Letters*, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865519301096

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

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

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, "A quantization-friendly separable convolution for mobilenets," 2018.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] W. Szpankowski, "Asymptotic average redundancy of huffman (and shannon-fano) block codes," in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, June 2000, pp. 370–.

[37] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and fisher pruning," *arXiv preprint arXiv:1801.05787*, 2018.

[38] E. Wang, J. J. Davis, P. Y. Cheung, and G. A. Constantinides, "Lutnet: Rethinking inference in fpga soft logic," *arXiv preprint arXiv:1904.00938*, 2019.

[39] A. W. Wijayanto, J. J. Choong, K. Madhawa, and T. Murata, "Towards robust compressed convolutional neural networks," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–8.

[40] Q. Xiao, Y. Liang, L. Lu, S. Yan, and Y.-W. Tai, "Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on fpgas," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. New York, NY, USA: ACM, 2017, pp. 62:1–62:6. [Online]. Available: http://doi.acm.org/10.1145/3061639.3062244

[41] Y. Xing, S. Liang, L. Sui, X. Jia, J. Qiu, X. Liu, Y. Wang, Y. Wang, and Y. Shan, "DNNVM : End-to-end compiler leveraging heterogeneous optimizations on fpga-based CNN accelerators," *CoRR*, vol. abs/1902.07463, 2019. [Online]. Available: http://arxiv.org/abs/1902.07463

[42] G. Yang, T. Zhang, P. Kirichenko, J. Bai, A. G. Wilson, and C. De Sa, "Swalp: Stochastic weight averaging in low-precision training," *arXiv preprint arXiv:1904.11943*, 2019.

[43] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 1916–1920.

[44] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *European Conference on Computer Vision (ECCV)*, 2018.