POLITECNICO DI TORINO
Repository ISTITUZIONALE

CoNLoCNN: Exploiting Correlation and Non-Uniform Quantization for Energy-Efficient Low-precision Deep Convolutional Neural Networks

Original
CoNLoCNN: Exploiting Correlation and Non-Uniform Quantization for Energy-Efficient Low-precision Deep Convolutional Neural Networks / Hanif, Muhammad Abdullah; Sarda, Giuseppe Maria; Marchisio, Alberto; Masera, Guido; Martina, Maurizio; Shafique, Muhammad. - ELETTRONICO. - 1:(2022), pp. 1-8. (Intervento presentato al convegno International Joint Conference on Neural Networks tenutosi a Padova, Italia nel 18-23 luglio 2022) [10.1109/IJCNN55064.2022.9892902].

Availability:
This version is available at: 11583/2972221 since: 2022-10-11T13:59:25Z

Publisher:
IEEE

Published
DOI:10.1109/IJCNN55064.2022.9892902

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

29 March 2024

© 2022 IEEE. This is the author's version of the work. The definitive Version of Record will be Published in the 2022 International Joint Conference on Neural Networks (IJCNN).

CoNLoCNN: Exploiting Correlation and Non-Uniform Quantization for Energy-Efficient Low-precision Deep Convolutional Neural Networks

Muhammad Abdullah Hanif [1,2], Giuseppe Maria Sarda [3], Alberto Marchisio [2],
Guido Masera [3], Maurizio Martina [3], Muhammad Shafique [1]
[1]*New York University Abu Dhabi,* Abu Dhabi, UAE
[2]*Technische Universität Wien (TU Wien)*, Vienna, Austria
[3]*Politecnico di Torino*, Turin, Italy
mh6117@nyu.edu, s255648@studenti.polito.it, alberto.marchisio@tuwien.ac.at,
guido.masera@polito.it, maurizio.martina@polito.it, muhammad.shafique@nyu.edu

arXiv:2208.00331v1 [cs.AR] 31 Jul 2022

***Abstract*—In today's era of smart cyber-physical systems, Deep Neural Networks (DNNs) have become ubiquitous due to their state-of-the-art performance in complex real-world applications. The high computational complexity of these networks, which translates to increased energy consumption, is the foremost obstacle towards deploying large DNNs in resource-constrained systems. Fixed-Point (FP) implementations achieved through post-training quantization are commonly used to curtail the energy consumption of these networks. However, the uniform quantization intervals in FP restrict the bit-width of data structures to large values due to the need to represent most of the numbers with sufficient resolution and avoid high quantization errors. In this paper, we leverage the key insight that (in most of the scenarios) DNN weights and activations are mostly concentrated near zero and only a few of them have large magnitudes. We propose CoNLoCNN, a framework to enable energy-efficient low-precision deep convolutional neural network inference by exploiting: (1) non-uniform quantization of weights enabling simplification of complex multiplication operations; and (2) correlation between activation values enabling partial compensation of quantization errors at low cost without any run-time overheads. To significantly benefit from non-uniform quantization, we also propose a novel data representation format, Encoded Low-Precision Binary Signed Digit, to compress the bit-width of weights while ensuring direct use of the encoded weight for processing using a novel multiply-and-accumulate (MAC) unit design.**

I. Introduction

Deep Neural Networks (DNNs) are state-of-the-art models for applications like object classification, image segmentation and speech processing [1]. However, they have high computational complexity and memory footprint, which translates to high hardware and energy requirements [2]. *This resource-hungry nature of DNNs challenges their high-accuracy deployment in resource-constrained scenarios such as inference on resource-constrained embedded devices.*

Methods like pruning and quantization are used to reduce the computational complexity and memory footprint of DNNs [3]–[6]. However, *these state-of-the-art approaches are highly effective when used with retraining* to minimize the accuracy loss. While retraining smaller DNNs designed for less complex datasets incurs fewer overheads, it is super costly to retrain larger DNNs designed for complex applications, which can take several days even on high-end GPU servers. Moreover, proper retraining of DNNs during optimization phase requires access to a comprehensive dataset, and may not be possible in cases where the dataset is an IP of a company and it has not made it available (e.g., Google's JFT-300M dataset [7]) to the end user who wants to optimize the given pre-trained DNN for a specific set of resource constraints. Under such cases, an effective approach is post-training quantization that enables low-precision Fixed-Point (FP) implementation ($\leq$8 bits) of DNNs [8] [9], and thereby reduces power, latency, and memory footprint. However, reducing precision introduces quantization errors, leading to potentially noticeable accuracy loss. *This additional source of error degrades the application-level accuracy of DNNs and restricts the designers from reducing the bit-widths of DNN datastructures beyond a level without significantly affecting the accuracy.*

State-of-the-Art Works and Their Limitations: To overcome the above-mentioned limitations and achieve significant efficiency gains, several works have been proposed. Compensated-DNN [10] proposed a technique to quantize DNN data-structures and dynamically compensate for quantization errors. It achieves this using a novel Fixed Point with Error Compensation (FPEC) data representation format and a specialized Processing Element (PE) design containing a low-cost error compensation unit. In FPEC, each number consists of two sets of bits: (1) bits that represent quantized value of the number in traditional FP formats; and (2) bits that store quantization related information, such as error direction, which is useful for error compensation. Techniques like [11] and [12] make use of power-of-two quantization and multi-scaled FP representation (respectively) to reduce the bit-width of DNN data-structures as well as the complexity of the PEs (mainly multipliers). Recently, CAxCNN [13] proposed the use of reduced precision Canonical-Sign-Digit (CSD) representation to decrease the complexity of Multiply-and-Accumulate (MAC) units in DNN accelerators. Similarly, [14] combined the use of reduced precision CSD with customized delta encoding scheme to efficiently approximate and encode DNN parameters. A summary of the key characteristics of the most relevant state-of-the-art techniques is presented in Table I. *The table shows that Compensated-DNN is the only work that employs error compensation (without retraining) to offer better energy-accuracy trade-off. However, it does not exploit the advantages of efficient data representations such as power-of-two or reduced precision CSD representations. Moreover, it also requires extra hardware support for error compensation.*

Other approaches that are also designed to improve the energy efficiency of DNN inference without the need of retraining involve the use of approximate hardware modules (mainly approximate multipliers) [15] [16], and voltage-scaling of computational array and memory modules [17] [18]. AxNN [15] selectively

TABLE I: Characteristics of the most relevant state-of-the-art works. RP-CSD refers to reduced precision CSD and T-CSD refers to truncated CSD representation.

	Compensated-DNN [10]	CAxCNN [13]	H.-P. Trinh et al. [14]	D. Liu et al. [9]	H. Tann et al. [11]	XNOR-Net [6]
Weights representation format	FPEC	RP-CSD	T-CSD	FP	Power-of-two	Binary
Involves retraining	No	No	No	No	Yes	Yes
Error compensation	Yes	No	No	No	-	-
Encoding of weights	No	No	Yes	No	Yes	No
Complementary hardware	Yes	No	Yes	No	Yes	No
Evaluation for ImageNet	Yes	Yes	No	Yes	Yes	Yes

approximates the neurons that do not impact the application-level accuracy much. However, it offers close to the baseline accuracy only with approximation-aware retraining. ALWANN [16] presents a method that employs functional approximations in the MAC units to improve the energy efficiency of DNN inference without involving approximation-aware retraining. It mainly determines the most suitable approximation for each layer of the given DNN to achieve a reasonable amount of savings. *However, note that these techniques have shown effectiveness only at 8-bit FP precision level (i.e., not for less than 8-bit precision) and only for relatively less complex datasets like Cifar-10.* Moreover, the voltage scaling techniques such as ThunderVolt [17] and MATIC [18] either introduce faults at run-time that can lead to undesirable accuracy loss or the need for expensive fault-aware retraining.

Summary of Key Challenges Targeted in this Work: Based on the above-mentioned limitations of the existing works, we highlight the following key challenges:

- There is a need to investigate existing DNN post-training quantization and approximation techniques to identify the methods and data representation formats that can enable high energy and performance efficiency without affecting the application-level accuracy of DNNs.
- To maintain high accuracy, dynamic error compensation requires additional resources that increase DNN inference cost. Therefore, there is a need to explore low-cost software-level error compensation mechanisms that are required to be applied only once during DNN conversion (i.e., design-time phase) and have the potential to offer benefits equivalent to costly dynamic error compensation techniques.

Our Novel Contributions: We propose an algorithm and architecture co-design framework, CoNLoCNN, for effectively approximating DNNs through post-training quantization to improve the power-/energy-efficiency of DNN inference process without involving retraining. Towards this:

1) We investigate different methods for quantizing DNN data-structures in search for an effective method for approximating DNNs that offers high energy-accuracy trade-off. We identify that choosing a data representation format that is aligned to the long-tailed data distribution of DNN parameters (see Fig. 3) results in less overall quantization error, and this selection can also help in simplifying the hardware components (mainly multipliers in the processing array of DNN hardware accelerators) (**Section III-A**). Based on our analysis, we propose an *Encoded Low-Precision Binary Signed Digit* data representation format that at the core is an evolved version of power-of-two representation and thereby enables the use of low-cost shift operations instead of multiplications. This results in significant simplification of the MAC units in DNN accelerators and thereby enables high energy and performance efficiency. (**Section IV**)
2) We propose a low-cost error compensation strategy that is required to be applied only once at the conversion-time (compile-time) to compensate for quantization errors. This enables the users to completely avoid overheads associated with dynamic compensation. (**Section III-B**)
3) We propose a systematic methodology for quantizing DNNs while exploiting the proposed error compensation strategy. (**Section V**)
4) We propose a specialized multiply-and-accumulate (MAC) unit design that fully exploits the potential of the proposed approximation scheme for improving the energy efficiency of DNN inference. (**Section IV-4**)

A summary of our novel contributions is shown in Fig. 1.

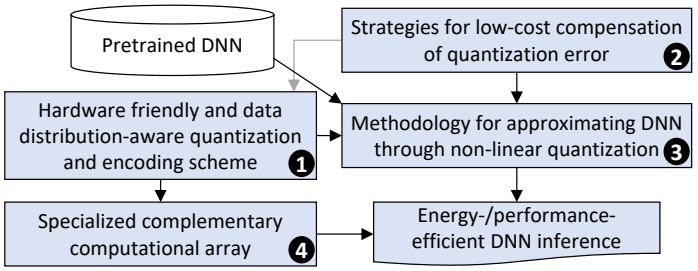


Fig. 1: System overview with our novel contributions and flow

II. Preliminaries

This section introduces the key terms used in this paper.

A. Deep Neural Networks (DNNs)

A DNN is an interconnected network of neurons, where each neuron performs a weighted sum operation, and then passes the output through a non-linear activation function. The functionality of a neuron can mathematically be written as $O = F(\sum_i W_i * A_i + b)$, where O represents the output, W_i represents the i^{th} weight, A_i represents the i^{th} input (a.k.a. activation), b represents the bias, and F represents the non-linear activation function. In DNNs, neurons are arranged in the form of layers. The layers are then connected in a specialized format to form a DNN.

B. Convolutional Neural Networks (CNNs)

A CNN is a type of DNN specialized for processing spatially correlated data such as images [1]. It is mainly composed of convolutional (CONV) layers and fully-connected (FC) layers (see Fig. 2(a)). The CONV layers are used to extract features from the input by convolving it with a set of filters (see Fig. 2(b)). The output generated by convolving a filter with the input is referred as a feature map. The FC layers are typically used at the end of the network for classification. A FC layer is composed of several neurons where each neuron receives the complete output vector of the previous layer to generate an output.

III. Strategies for Enabling Low-Precision and Energy-Efficient DNN Inference

A. Strategy 1: Select a quantization scheme that is aligned with the data distribution

Fig. 3 shows the distributions of weights, biases, and activations of the layers of a trained AlexNet. It can be observed from the figure that the distributions of weights and activations

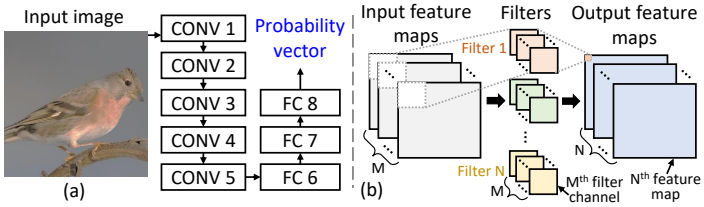
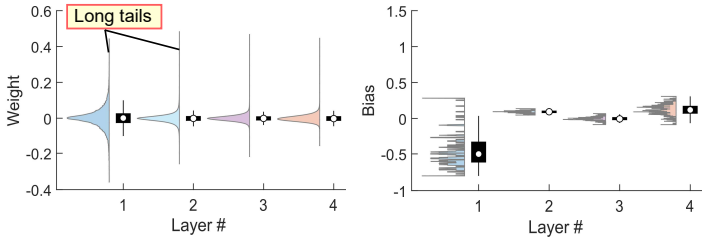
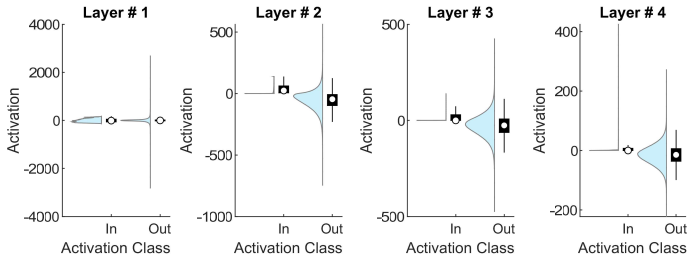


Fig. 2: (a) Architecture of AlexNet. (b) Illustration of a CONV.

have long tails, i.e., in each data-structure, majority of the values have small magnitude and only a limited number of values have large magnitude. Moreover, for each layer, the distribution of weights is close to a Gaussian distribution with mean equal to zero. Considering the data distributions, a low precision uniform quantization (for example, see Fig. 4(a)) would result in high overall quantization error compared to a non-uniform quantization (for example, see Fig. 4(b)) having the same (or less) number of quantization levels, assuming the levels are distributed based on the data distribution, i.e., more number of narrowly-spaced quantization levels in dense regions and less number of widely-spaced quantization levels in light regions. Therefore, aligning quantization scheme with the data distribution can help in reducing the overall/average quantization error. However, a potential limitation of this is that, in case of low-precision non-uniform quantization, it can result in a significant increase in the maximum quantization error leading to high error variance. *Hence, the ideal quantization scheme should balance between the average and the maximum quantization error to achieve minimum quality degradation.*



(a) Distribution of weights of the layers (b) Distribution of biases of the layers



(c) Distribution of input and output activations of the layers

Fig. 3: Distribution of weights, biases and activations of the first four convolutional and layers of the AlexNet (in the form of half-violin plots and box plots). Note that the output activations here represent the output of the layer before passing through activation functions.

B. Strategy 2: Exploit correlation between neighboring feature map values to reduce the effective variance and mean of quantization error

Here, we first analyze the impact of variations in the bias values of a CNN on its classification accuracy. Note that this analysis mainly helps us understand the effects of errors that affect the mean of activation values. Then, we study the correlation of data within and across input feature maps of a layer to see if quantization errors can be partially compensated by distributing them across weights in the same filter/neuron. Afterward, we present a mathematical analysis and show how the gained insights can be exploited to reduce the impact of quantization errors.

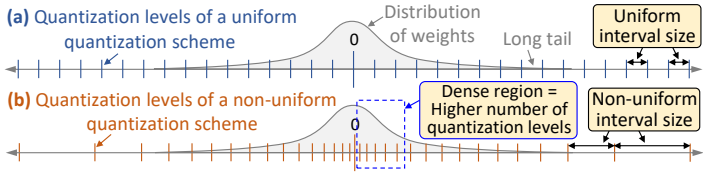


Fig. 4: Comparison between uniform and non-uniform quantization.

1) Impact of modifying the bias values in DNNs: Fig. 5 shows the impact of varying the bias of different number of filters of a layer of a pre-trained AlexNet on its classification accuracy. From Fig. 5(a) and Fig. 5(c) it can be observed that, when a small constant value, i.e., a value close to the range of original bias values (see Fig. 3(b)), is added to the bias values of a number of filters, the accuracy of the network stays close to its baseline. However, when the magnitude of the constant is large, it degrades the accuracy. The difference between the impact of positive and negative noise is mainly due to the presence of ReLU activation function in the AlexNet, as a large negative bias leads to a large negative output which is then mapped to zero by the ReLU function and thereby limits the impact of error on the final output. Fig. 5(b) and Fig. 5(d) show that, when the bias values of half of the filters are injected with positive noise and half with negative noise (all having the same magnitude), the behavior of classification accuracy is dominated by the behavior of filters that are injected with positive noise.

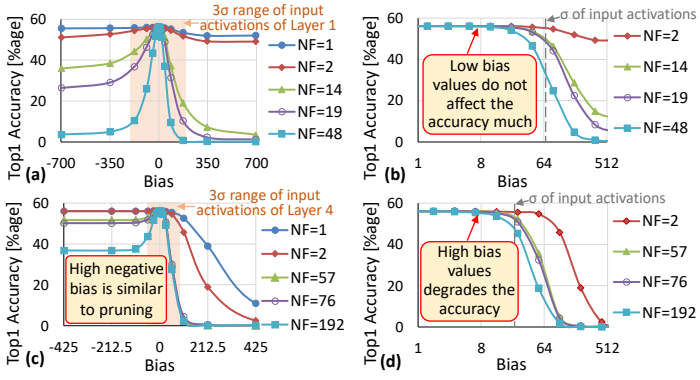


Fig. 5: Impact of altering the bias values of different number of randomly selected filters/neurons (NF) of different layers of a trained AlexNet (trained on the ImageNet dataset) on its classification accuracy. (a) and (c) show the impact when same amount of positive (or negative) value is added to the bias values of the selected filters of layer 1 and 4, respectively. (b) shows the impact when the bias values of half of the selected filters of layer 1 are injected with positive noise and half with negative noise having the same magnitude. Similar to (b), (d) shows the results for layer 4.

To further study the impact of mean shifts in the output feature maps, we performed an experiment where we added noise

generated using a Gaussian distribution to the bias values of the filters/neurons of different layers of a pre-trained AlexNet (see Fig. 6). We observed that when the noise is generated using smaller standard deviation values and is injected to the intermediate layers of the network, it does not impact the accuracy much. However, if the noise is injected to the last layer of the network or is generated using larger standard deviation values, it leads to a significant drop in the DNN accuracy.

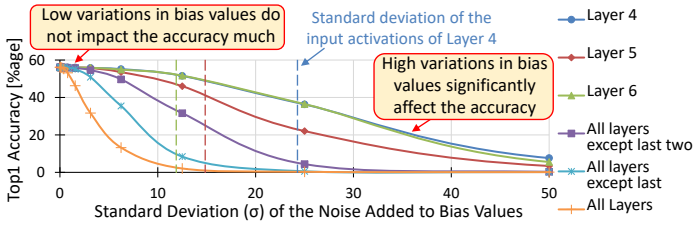


Fig. 6: Impact of adding noise generated using a Gaussian distribution to the bias values of filters/neurons of different layers and different number of layers of a trained AlexNet on its classification accuracy.

From the above analysis, we deduce the following three conclusions. *(1) Small to moderate level of variations due to any error/noise source in the mean of activation values of all the layers except the last layer do not impact the accuracy much. (2) Mean shift in the output degrades the accuracy only if it is large in magnitude or it is in the output of last layer of a DNN. (3) Resilience of DNNs to small-to-moderate level of errors in bias values points to the significance of large activation values.*

2) Correlation between activation values of input feature maps: Intra-Feature Map Correlation: Fig. 7 shows the correlation between neighboring input activation values located at a constant shift from each other (represented using i and j in the figure) within input feature maps of a convolutional layer. Based on the correlation values in the figure, it can be said that, in all the convolutional layers of a DNN, there is a significant correlation between the neighboring activation values.

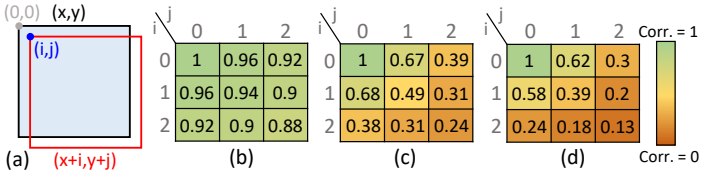


Fig. 7: Intra-feature map correlation of input activations of different layers of the AlexNet and the VGG16. (a) Illustration of an input feature map (shown in blue) and its shifted variant (shown with red border). i and j define the shifts in x and y directions, respectively. (b) and (c) show the correlation between the input feature maps and their shifted variants of layer 1 and layer 3 of the AlexNet, respectively. (d) shows correlation between neighboring input activations of layer 12 of the VGG16.

Inter-Feature Map Correlation: Fig. 8 shows the distribution of correlation between different input feature maps of a layer of a pre-trained AlexNet. Fig. 8(a) shows that there is a significant correlation between the input feature maps of layer 1 of the network. The distributions in Figs. 8(b), 8(c), and 8(d) show that as we move deeper into the network the across-feature map correlation moves towards zero.

Based on the above analysis, we conclude that only intra-feature map correlation can be exploited for error compensation.

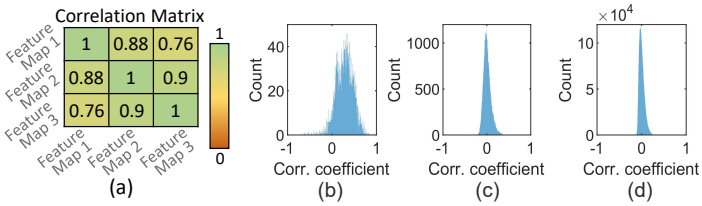


Fig. 8: Correlation between input feature maps of different layers of the AlexNet. (a) Correlation matrix of input feature maps of layer 1. (b) Distribution of the correlation between input feature maps of layer 2. Similar to (b), (c) and (d) show distribtions of layer 4 and layer 7, respectively.

3) Analysis of quantization error: To analyze the effects of quantization on the output quality, let us consider a scenario in which we quantize the weights of a layer of a DNN and keep the activations in full-precision format. A quantized weight can be written as

$$W_q = W + \Delta W \tag{1}$$

where W represents unquantized weight and ΔW represents quantization error. If we assume $W \sim \mathcal{N}(\mu_W, \sigma_W^2)$ and $\Delta W \sim \mathcal{N}(\mu_{\Delta W}, \sigma_{\Delta W}^2)$, and W and ΔW to be independent, then

$$W_q \sim \mathcal{N}(\mu_W + \mu_{\Delta W}, \sigma_W^2 + \sigma_{\Delta W}^2) \tag{2}$$

Similar to the distribution of W, for activations, we assume

$$A \sim \mathcal{N}(\mu_A, \sigma_A^2) \tag{3}$$

Now, for $O_q = \sum_{i=1}^{n} W_{qi} * A_i$, using the above equations and assuming the weights and activations to be independent, we get

$$O_q \sim \mathcal{N}(n * \mu_A * (\mu_W + \mu_{\Delta W}),\ n * ((\sigma_W^2 + \sigma_{\Delta W}^2 + (\mu_W + \mu_{\Delta W})^2) * (\sigma_A^2 + \mu_A^2) - \mu_A^2 * (\mu_W + \mu_{\Delta W})^2)) \tag{4}$$

As highlighted in the earlier analysis, small deviations in the mean of output activations do not impact the accuracy much; therefore, we mainly focus on comparing the variance of O_q with the variance of $O = \sum_{i=1}^{n} W_i * A_i$. Subtracting variance of O from the variance of O_q, we are left with

$$n*(\sigma_{\Delta W}^2*\sigma_A^2+\sigma_{\Delta W}^2*\mu_A^2+\sigma_A^2*\mu_{\Delta W}^2+2*\sigma_A^2*\mu_W*\mu_{\Delta W}) \tag{5}$$

Now, to reduce the intensity of this additional term, we need to reduce the intensity of $\sigma_{\Delta W}$ and $\mu_{\Delta W}$. We can achieve this by exploiting the high correlation among the neighboring activation values in feature maps. Fig. 9(a) shows a possible way of decomposing activation values of a block of feature map based on correlation among neighboring values. In case of high correlation (for example, see Fig. 9(b)), the variables x, y and z (in Fig. 9(a)) have high values (i.e., close to 1). As $\bar{a_2}$, $\bar{a_3}$ and $\bar{a_4}$ represent the elements of a_2, a_3 and a_4 (respectively) that are orthogonal to a_1, the variables $\bar{a_2}$, $\bar{a_3}$ and $\bar{a_4}$ exhibit low variance compared to a_2, a_3 and a_4 in the case of high correlation with a_1. We can exploit the presence of xa_1, ya_1 and za_1 in the

neighboring activations to partially compensate/balance the error introduced in the dot-product of weights and activations due to quantization of weights by modifying the quantization scheme in such a way that it balances the mean quantization error of the neighboring weights.

Example: To understand this, consider the activation block A shown in Fig. 9(c) and 2D filter W shown in Fig. 9(d). The output of dot-product of A and W comes out to be 50.32. Now, if we quantize the weights of the filter to the nearest integer values and perform the dot-product operation, we get 57.7 as the output; see Fig. 9(e). However, if we map the value of the second weight to its other nearest integer value, i.e., 2 instead of 3 (see Figs. 9(e) and 9(f)), we can reduce the mean absolute error (MAE) in weights from 0.225 to 0.025 and the absolute error in the output of dot-product from 7.38 to 1.12. This shows that high correlation among the neighboring values enable us to reduce effective mean and variance of ΔW by minimizing the local mean quantization error inside filter channels.

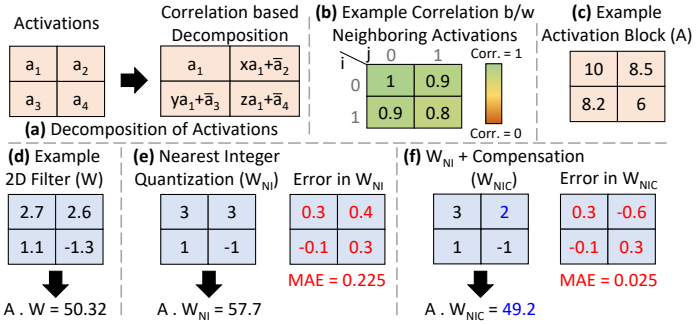


Fig. 9: Decomposition of activations, and an example illustrating the impact of exploiting correlation for error compensation on the output of dot-product operation.

4) Impact of Adjusting Intra-channel Mean Quantization Error in Weights: To study the impact of adjusting the mean error in filters, we performed an experiment where we injected noise generated using a Gaussian distribution to the weights of the filters of layer 1 and layer 4 of the AlexNet. We studied three different cases: (1) No adjustment in the mean error of the weights; (2) Mean error adjustment case 1, where the mean error of each filter is subtracted from the corresponding weights; and (3) Mean error adjustment case 2, where the mean error of each filter channel is subtracted from the corresponding weights. *Fig. 10 shows that, among the three cases, intra-channel adjustment (i.e., Mean error adjustment case 2) leads to highest compensation and thereby best results.*

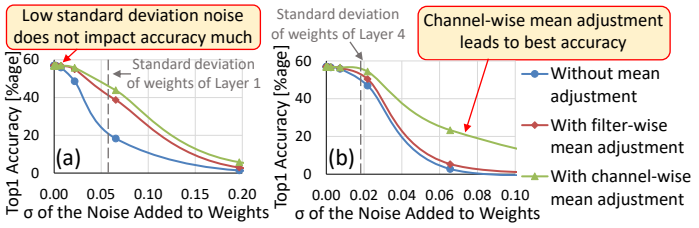


Fig. 10: Impact of adjusting the inter- and intra-channel mean quantization error in the weights of AlexNet. (a) Layer 1; (b) Layer 4.

IV. Type of Non-Uniform Quantization and Design of Supporting DNN Hardware

Strategy 1 in Section III states that aligning quantization scheme with the data distribution helps in restricting the overall quantization error. However, the key challenge is *how can this observation be exploited for improving the efficiency of a DNN-based system.* To address this, we propose *Encoded Low-Precision Binary Signed Digit (ELP_BSD)* representation that evolves from power-of-two representation and thereby enables the use of shift operations instead of multiplications in processing arrays of DNN accelerators. In the following, we discuss the details of our data representation format, starting with the initial concept, limitations of initial proposition, and how ELP_BSD overcomes these limitations.

1) Initial Proposition: In this work, the key focus is on exploiting non-uniform quantization for simplifying the MAC unit design as well as in reducing the bit-width of weights, as both contribute towards improving the energy-efficiency. Power-of-two quantization is one potential solution, as it allows to replace a costly multiplication operation with a shift operation and reduce the bit-width by storing only the power of 2. Moreover, the distribution of quantization levels of power-of-two quantization is aligned with the distribution of DNN weights, as can be observed from Fig. 11(b). However, use of power-of-two quantization results in a significant drop in application-level accuracy due to a significant reduction in the number of quantization levels compared to a traditional FP quantization, which can be observed by comparing Fig. 11(b) with Fig. 11(a). *Therefore, almost all the previous works that use power-of-two quantization employ retraining to re-gain the lost accuracy.* To offer additional quantization levels to avoid significant accuracy loss due to quantization while benefiting from the advantages of power-of-two quantization scheme, we propose to use sum of signed power-of-two quantization, where more than one signed power-of-two digits are combined to offer additional quantization levels. Fig. 11(c) shows that addition of only a single low-range low-precision signed power-of-two digit can significantly increase the number of unique quantization levels and thereby can help in achieving an ultra-efficient system.

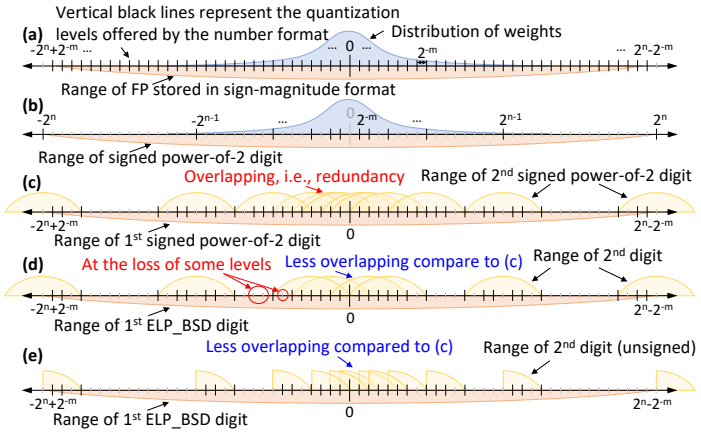


Fig. 11: Illustration of different data representation formats that show step-by-step evolution of traditional quantization scheme to our ELP_BSD representation.

2) Limitations of Initial Proposition and New Improvements: One key issue with sum of signed power-of-two quantization is

redundant representations of values. For example, $2^{x_1} - 2^{x_2} = 0 \; \forall \; x_1 = x_2$. To reduce the amount of redundancy, we propose to reduce the number of possible power of 2 values per digit. For example, if a number representation is given as $2^{x_1} - 2^{x_2}$ and $x_1, x_2 \in \{0, 1, 2, 3\}$, to reduce the redundancy, we can reduce the set of possible values of x_1 to $\{1, 3\}$. Fig. 11(d) shows that reducing the number of possible power of 2 values for the first digit leads to a reduction in the amount of redundancy, which can be observed from the reduced number of yellow semi-circles (representing the range of the second digit) centered at every possible quantization level of the first digit. Note that the reduction in redundancy can be exploited to reduce the bit-width of numbers as well as to further simplify the MAC unit design. However, a drawback of this is that it can result in a small decrease in the number of quantization levels, as highlighted in Fig. 11(d). The redundancy can also be reduced by allowing some digits to have signed values and some only positive (i.e., unsigned) values. This is illustrated in Fig. 11(e) by restricting the range of the second digit to only positive values. *Note that even though the sum of power-of-2 digits leads to some redundancy, it enable us to use shifters instead of multipliers in the hardware, which significantly improves the energy efficiency of DNN systems (as will be highlighted in Section VI). Moreover, the above explanation shows that the redundancy can be controlled by intelligent selection of the exact number representation format.*

3) ELP_BSD data representation: To efficiently represent low-precision sum of signed power-of-two numbers, we define a novel data representation format, Encoded Low-Precision Binary Signed Digit (ELP_BSD) representation. Fig. 12(a) shows how the specifications of a ELP_BSD representation are defined, and Fig. 12(b) shows the corresponding binary representation format. As shown in Fig. 12(b), the bits are divided into m groups, where each group is responsible for representing a single signed power-of-two digit. Each group consists of a sign bit and $ceil(log_2(n_i))$ bits to represent the index of shift count in the digit specification, where n_i is the number of different shift counts mentioned for i^{th} digit in the specification. Note that the sign bit is optional and only used when the corresponding digit is signed. Fig. 12 also presents two examples to explain the conversion of ELP_BSD numbers to FP values.

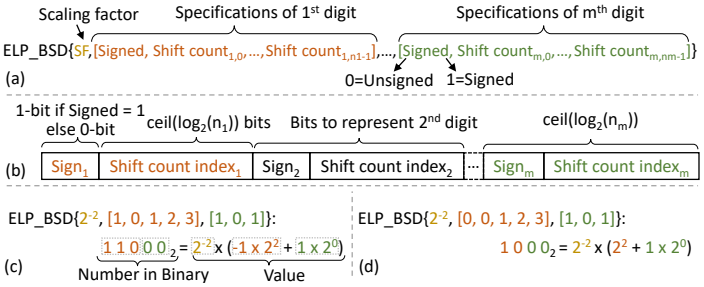


Fig. 12: (a) Specifications of an ELP_BSD format. (b) ELP_BSD format. (c) and (d) Examples to explain conversion between ELP_BSD format and values.

4) Supporting Hardware Design: As shown in Fig. 12, ELP_BSD format mainly stores power-of-two digits in an encoded format. To efficiently implement multiplication with a power-of-two digit at hardware-level, shifters (e.g., a barrel shifter) can be used. To realize a MAC unit, the shifter is followed by an adder that adds previously computed partial sum to the newly computed product. Fig. 13(a) shows the MAC unit design for the case when weights are represented using a single signed power-of-two digit. If we use the same ELP_BSD format for all the weights, we can hard code the indexing functionality in the shifter and use indexes directly from the encoded weight for multiplication. Moreover, we can choose the set of possible shift counts in a manner that it results in a less complex shifter design.

In case weights are represented using an ELP_BSD format that contains multiple digits, we can use multiple of these units (one per-digit) in parallel to compute the partial sums, which then have to be added together to generate one output. To achieve this addition, we propose to use a compressor tree followed by a multi-bit adder to add the outputs of the shifters and the partial sum from the previous computation to generate only a single output. Fig. 13(b) shows how multiple single digit MAC units can be integrated in the Processing Elements (PEs) of a Neural Processing Array (NPU), e.g., like the Tensor Processing Unit (TPU) [19]. Fig. 13(c) shows the processing array design of the TPU like architecture. This processing array follows a weight stationary dataflow where weights are kept stationary inside the PEs during execution. The input activations are fed from the left and moved towards the right over clock cycles. Similarly, the partial sums are moved towards the bottom of the array. Note that, for this work, we choose to represent activations using FP and 2's complement format, as changing the format of activations won't have any significant impact on the length of the adders in PEs. It is also important to highlight here that in most of the case 1-3 single digit MAC units per PE are sufficient to meet the user-defined accuracy constraints.

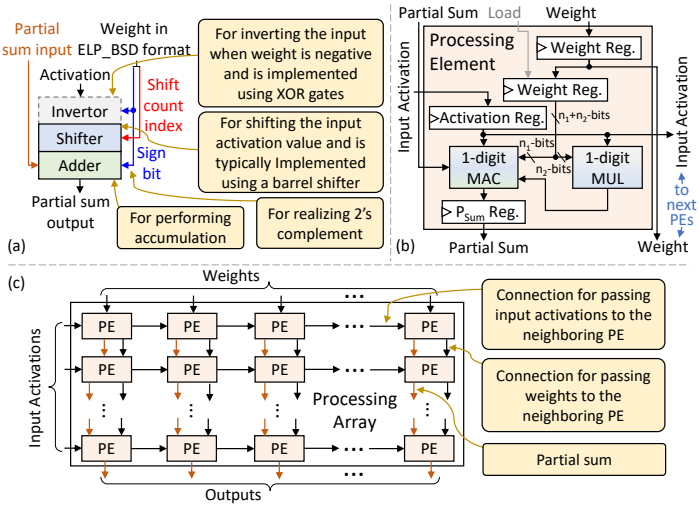


Fig. 13: (a) Single digit MAC design. (b) Modified processing element for an NPU. (c) A Neural Processing Array architecture.

V. Our Methodology for Efficient Approximation of CNNs through Non-Uniform Quantization

Fig. 14 presents our methodology for approximating CNNs through non-uniform quantization while exploiting the strategies mentioned in Section III. The following steps explain working of the methodology.

1) **Determine critical FP bit-width of activations:** Starting from maximum allowed FP bit-width activations, we gradually reduce the bit-width to find the critical point after which the accuracy loss of the input DNN raises above the

user-defined accuracy loss constraint (AC). For this step, we assume bit-width of all the activations to be the same. The key intuition behind this step is that a decrease in the activations' bit-width results in a linear decrease in the width of MAC units, which can help in improving the energy-efficiency. This step outputs critical bit-width for activations, denoted as CBW_A.

2) **Determine scaling factor for weights:** Given the data representation format for weights, we compute the scaling factor for weights of each layer of the input DNN separately. For this work, in case of ELP_BSD format, we compute the scaling factor as $SF = max(weights)/2^{max(shift_counts)}$.
3) **Apply nearest neighbor quantization:** Using the data representation format and the scaling factors, we generate a table of possible quantization levels (TQL) for each layer of the input DNN. Then to perform quantization, we replace the weights with their corresponding nearest values in the tables.
4) **Apply error compensation algorithm:** For each convolutional layer, we pass the weights to Algo. 1, which (partially) compensates for errors introduced due to quatization of weights by exploiting Strategy 2 mentioned in Section III. Note that as most of state-of-the-art architectures use small filter sizes, e.g., 3x3, Algo. 1 focuses on compensating the overall channel-level mean quantization error in filter weights. To achieve this, it computes the mean quantization error of a channel of a filter, locates the values that can be mapped to their other neighboring quantization level to reduce the mean error, sorts all the located values based on a cost function and starts altering the values starting form the values having least cost till the point the absolute mean error is decreasing.
5) **Estimate the overall accuracy loss:** In this step, we compute the accuracy to check if the user-defined accuracy constraint is met. In case the constraint is not satisfied, the algorithm increases the value of CBW_A by 1 and performs accuracy evaluation again. If the constraint is still not met and CBW_A becomes equal to BW_{max}, it outputs the latest quantized DNN.

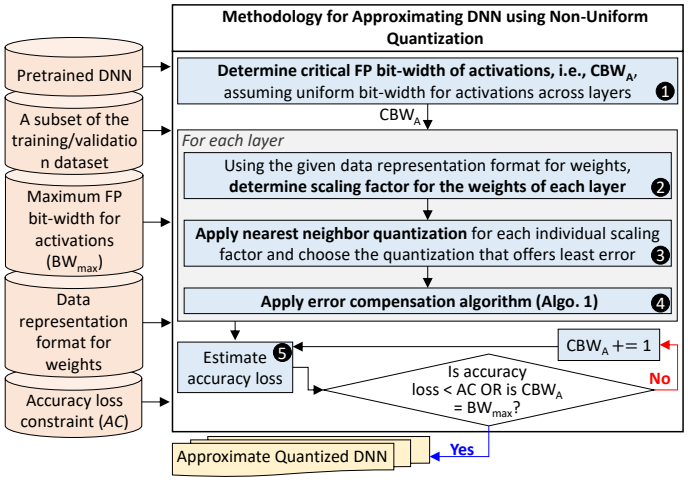


Fig. 14: Our DNN quantization methodology

VI. Results and Discussion

A. Experimental Setup

To evaluate CoNLoCNN, we extended MatConvNet [20] framework for FP implementation and our CoNLoCNN methodology. We evaluated CoNLoCNN using two popular DNNs used for benchmarking FP implementations, i.e., AlexNet and VGG-16 trained on the ImageNet dataset. For hardware synthesis, we implemented PEs composed of different MAC unit designs in Verilog and synthesized for the TSMC 65nm technology using Cadence Genus.

Algorithm 1: Pseudo-code for error compensation

Input: Un-quantized weights of a CONV layer ($UnQunat_W$); Table of possible quantization levels (TQL)
Result: Quantized weights ($Quant_W$)
$Quant_W \leftarrow$ NNQuat($UnQunat_W$, TQL); % Nearest neighbor quantization
$Error_W \leftarrow UnQunat_W - Quant_W$; % Error in weights
for ($i = 1$; $i \leq$ *No of filters* ; $i+ = 1$) **do**
  for ($j = 1$; $i \leq$ *No of channels* ; $j+ = 1$) **do**
    $Mean_Err$ = mean($Error_W(:, :, i, j)$); % Mean error of the channel
    $S \leftarrow$ Subset of $UnQunat_W(:, :, i, j)$ having corresponding error opposite in sign to $Mean_Err$
    $SO \leftarrow$ Values of S quantized to closet levels in the opposite direction to the nearest neighbor
    $Cost_S = abs(S - SO)$
    $Sorted_S \leftarrow$ Set of sorted values of S in order of increasing cost
    for ($k = 1$; $i \leq$ *No of values in* $Sorted_S$; $k+ = 1$) **do**
      $New_{Mean}Err \leftarrow$ Mean quantization error if the quantized value of $Sorted_S(k)$ in $Quant_W$ is replaced with the corresponding value from SO
      if *abs(Mean_Err) ¿ abs(New_Mean_Err)* **then**
        Accept the change in quantization level of value corresponding to $Sorted_S(k)$ in $Quant_W$
        $Mean_Err = New_Mean_Err$
      else
        break
      end
    end
  end
end

B. Effectiveness of Our Error Compensation Strategy, i.e., Algorithm 1

To demonstrate the effectiveness of error compensation algorithm, we applied it with FP implementation and compared the results with conventional FP implementation. Note that for this experiment, we assumed the bit-width of weights and activations to be the same and uniform across all the layers of the DNN. Fig. 15(a) shows the results for the AlexNet. As shown in the figure, our error compensation strategy helps in improving the accuracy of the network, specifically at lower bit-widths.

C. Effectiveness of CoNLoCNN for State-of-the-Art DNNs

To demonstrate the effectiveness of our overall methodology, we considered four different ELP_BSD data representation formats. The formats are listed in Table II along with the hardware characteristics of the PEs implemented using the corresponding MAC designs for a 32x32 processing array (shown in Fig. 13(c)). Note that the scaling factor (represented by x in the table) of the representations is not considered to be the same across layers, and it is selected based on the statistics of the parameters of the corresponding layer. For each representation, we considered five different activation bit-widths, i.e., 8-bit till 4-bit, to study the impact of change in activation bit-width on the accuracy of the network and the hardware efficiency. Fig. 15(b) shows the

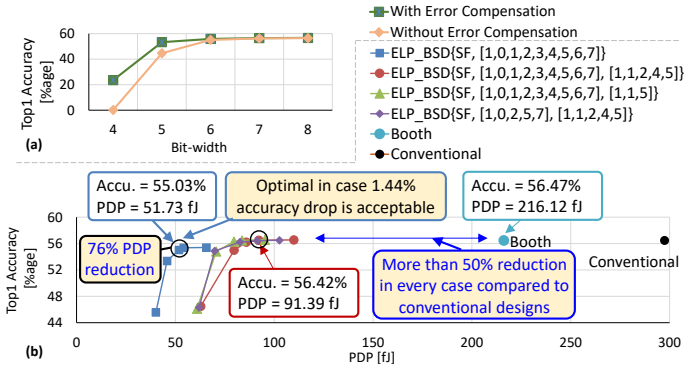


Fig. 15: (a) Effectiveness of our error compensation strategy when used with traditional FP quatization for the AlexNet. (b) Accuracy of the AlexNet vs. PDP for different ELP_BSD data representations.

accuracy vs. PDP results achieved when CoNLoCNN is used for the AlexNet considering the ELP_BSD configurations mentioned in Table II. The plots on the left inside Fig. 15(b) are generated by CoNLoCNN while the two points on the right represent conventional designs considered for comparison. The plot shows that as the bit-width of the activations decreases, we observe a slight decrease till a point after which the rate of decay increases drastically. Moreover, different ELP_BSD formats offer different accuracy-efficiency characteristics. *The key thing to observe in Fig. 15(b) is that even the most power consuming PE design generated by the proposed CoNLoCNN framework offers around 50% reduction in PDP compared to conventional designs.* In case 1.44% drop in accuracy is acceptable, the proposed method can offer around 76% reduction in the PDP. Similar results are observed for the VGG-16 network.

TABLE II: Hardware characteristics of the PEs designed using our methodology for some of ELP_BSD representations and their comparison with booth multiplier-based and conventional multiplier-based PEs.

	Bit-width		With 8-bit Activations			
Configuration of MAC for PEs	Weights	Activations	Area [cell area]	Power [uW]	Delay [ns]	PDP [fJ]
ELP_BSD{x, [1,0,1,2,3,4,5,6,7]}	4	8	556	28.55	2.30	65.68
		5	450	23.06	1.99	45.79
ELP_BSD{x, [1,0,1,2,3,4,5,6,7], [1,1,2,4,5]}	7	8	838	59.60	1.85	109.96
		5	694	46.53	1.71	79.71
ELP_BSD{x, [1,0,1,2,3,4,5,6,7], [1,1,5]}	6	8	814	51.65	1.85	95.29
		5	676	41.22	1.71	70.65
ELP_BSD{x, [1,0,2,5,7], [1,1,2,4,5]}	6	8	835	56.57	1.81	102.61
		5	680	43.07	1.62	69.86
Booth Multiplier-based MAC	8	8	1195	86.73	2.49	216.12
Conventional FP	8	8	1179	83.56	3.56	297.47

D. Comparison with the state-of-the-art

To compare the results with the state-of-the-art, we implemented CAxCNN [13] inside our framework. We selected Canonical Approximate (CA) representation with 1 non-zero digit, where the weights are converted using their exhaustive search algorithm (i.e., their best algorithm) to have a fair comparison. For AlexNet, CAxCNN with 1 non-zero digit CA representation achieves 50.9% top1 accuracy while CoNLoCNN achieves 55.4% accuracy (i.e., close to the baseline). This improvement is mainly due to our error compensation strategy. The quantization levels offered by 1 non-zero digit CA are almost the same as offered by ELP_BSD{SF,[1,0,1,2,3,4,5,6,7]} format with the only difference being of '0', which is not present in ELP_BSD{SF,[1,0,1,2,3,4,5,6,7]}. However, absence of '0' does not affect the accuracy due to the presence of 1 and -1 quantization levels and the use error compensation. Note that the absence of '0' in the highlighted case helps in achieving a simplified PE design. Even in the best possible scenario, CA representation would require 5 bits per weight while ELP_BSD{SF,[1,0,1,2,3,4,5,6,7]} requires 4 bits per weight. Similar to the AlexNet, for the VGG-16, we observed 3% higher accuracy with CoNLoCNN compared to CAxCNN. Note that CoNLoCNN not only helps in reducing the complexity of the hardware but also helps in reducing the memory footprint of DNNs unlike other approximate computing works (e.g., [16]) that operate at 8-bit precision.

VII. Conclusion

In this paper, we proposed CoNLoCNN, a framework to enable energy-efficient low-precision approximate DNN inference. CoNLoCNN mainly exploits non-uniform quantization of weights to simplify processing elements in the computational array of DNN accelerators and correlation between activation values to (partially) compensate for the quantization errors without any run-time overheads. We also proposed, Encoded Low-Precision Binary Signed Digit (ELP_BSD) representation, to reduce the bit-width of weights while ensuring direct use of the encoded weight in computations by designing supporting MAC units.

Acknowledgment

This research is partly supported by the ASPIRE AARE Grant (S1561) on "Towards Extreme Energy Efficiency through Cross-Layer Approximate Computing".

References

[1] Y. LeCun et al., "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] V. Sze et al., "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
[3] S. Anwar et al., "Structured pruning of deep convolutional neural networks," *ACM JETC*, vol. 13, no. 3, pp. 1–18, 2017.
[4] S. Han et al., "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[5] C. Zhu et al., "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.
[6] M. Rastegari et al., "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.
[7] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.
[8] P. Gysel et al., "Hardware-oriented approximation of convolutional neural networks," *arXiv preprint arXiv:1604.03168*, 2016.
[9] D. Lin et al., "Fixed point quantization of deep convolutional networks," in *ICML*, 2016, pp. 2849–2858.
[10] S. Jain et al., "Compensated-dnn: energy efficient low-precision deep neural networks by compensating quantization errors," in *ACM/ESDA/IEEE DAC*, 2018, pp. 1–6.
[11] H. Tann et al., "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *ACM/EDAC/IEEE DAC*, 2017, pp. 1–6.
[12] S. Jain et al., "Biscaled-dnn: Quantizing long-tailed datastructures with two scale factors for deep neural networks," in *ACM/IEEE DAC*, 2019, pp. 1–6.
[13] M. Riaz et al., "Caxcnn: Towards the use of canonic sign digit based approximation for hardware-friendly convolutional neural networks," *IEEE Access*, vol. 8, pp. 127 014–127 021, 2020.
[14] H.-P. Trinh et al., "Efficient data encoding for convolutional neural network application," *ACM TACO*, vol. 11, no. 4, pp. 1–21, 2015.
[15] S. Venkataramani et al., "Axnn: Energy-efficient neuromorphic systems using approximate computing," in *IEEE/ACM ISLPED*, 2014, pp. 27–32.
[16] V. Mrazek et al., "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," *arXiv preprint arXiv:1907.07229*, 2019.
[17] J. Zhang et al., "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *IEEE DAC*, 2018, pp. 1–6.
[18] S. Kim et al., "Matic: Learning around errors for efficient low-voltage neural network accelerators," in *IEEE DATE*, 2018, pp. 1–6.
[19] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *IEEE ISCA*, 2017, pp. 1–12.
[20] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.