Exploiting Resiliency for Kernel-wise CNN Approximation enabled by Adaptive Hardware Design

Cecilia De la Parra*[1], Ahmed El-Yamany*[1], Taha Soliman*, Akash Kumar†, Norbert Wehn‡, Andre Guntoro*,
*Robert Bosch GmbH, Renningen, Germany. {cecilia.delaparra, ahmed.el-yamany, taha.soliman, andre.guntoro}@de.bosch.com
†Technological University of Dresden, Dresden, Germany. akash.kumar@tu-dresden.de
‡University of Kaiserslautern, Kaiserslautern. wehn@eit.uni-kl.de

***Abstract*—Efficient low-power accelerators for Convolutional Neural Networks (CNNs) largely benefit from quantization and approximation, which are typically applied layer-wise for efficient hardware implementation. In this work, we present a novel strategy for efficient combination of these concepts at a deeper level, which is at each channel or *kernel*. We first apply layer-wise, low bit-width, linear quantization and truncation-based approximate multipliers to the CNN computation. Then, based on a state-of-the-art resiliency analysis, we are able to apply a kernel-wise approximation and quantization scheme with negligible accuracy losses, without further retraining. Our proposed strategy is implemented in a specialized framework for fast design space exploration. This optimization leads to a boost in estimated power savings of up to 34% in residual CNN architectures for image classification, compared to the base quantized architecture.**

***Index Terms*—Resiliency, Kernel-wise optimization, Approximate Computing, CNN inference, AI Accelerator.**

I. Introduction

CNNs are currently the key building blocks in several emerging applications which require fast inference execution, ultra-low power and high reliability. Two main approaches have been proposed in the literature to meet these requirements: CNN quantization and hardware approximation [1]–[3].

In this work, we combine the concepts of mapping quantization error to approximation, and using resiliency metrics to perform kernel-wise optimization. Resiliency analysis for CNN safety optimization has been explored in [4]–[6], where resiliency metrics were computed using Taylor expansion [7] to identify more sensitive kernels and propose a more reliable design architecture. A similar approach is presented in [8] where the more relevant kernels, identified by layer-wise Relevance Propagation [9], are computed in more resilient hardware cores. The use of resiliency for CNN approximation has also been explored in [10]–[12], where authors use different resiliency metrics for neuron-wise approximation. Drawbacks from these approaches are the challenging implementation in hardware of the obtained configurations, and their limitation to small CNN architectures. In this context, we demonstrate a favorable trade-off between approximation level and hardware complexity.

Methodologies that combine CNN quantization and approximation have been presented in [13], [14]. While both approaches were used for CNN optimization, we extend the state of the art by exploring the further combination of both at a kernel-wise level, which leads to improved energy savings with minor accuracy losses.

For CNN approximation, we select truncated multipliers without error correction. In [15], a truncation-based approximate multiply-and-accumulate (MAC) unit architecture with error compensation was introduced. Our proposed approximation approach, on the other hand, does not require a compensation function in hardware, as the introduced truncation error is leveraged at algorithmic level. This results in no area and power overhead from introducing the truncated multiplier.

In summary, our contributions are:

- A novel approach to efficiently combine CNN quantization and approximation at layer and kernel level.
- A methodology for design space exploration to effectively select and approximate kernels based on their relevancy metric, with negligible accuracy loss.
- Hardware realizations on both single MAC unit level as well as system level.

Our approach yields energy efficient CNNs with minimum accuracy loss. To our best knowledge, this is the first work to explore kernel-wise CNN approximation and quantization. Our experimental results show that our proposed optimization can achieve up to 34% further power savings in three different Residual architectures [16] trained with CIFAR10 [17].

II. Preliminaries

A. Convolutional Neural Networks

CNN architectures are primarily built of convolutional layers. Their function is to extract relevant features by convolving a weight tensor W, parameterized by a height H_w, width W_w, depth Ch_{in} and channels or *kernels* Ch_{out} with an input tensor X of shape H_{in}, W_{in}, C_{in} and adding kernel bias b_k. The output of a convolutional layer is computed by:

$$Y_{k,l,n} = \sum_{i,j,p} \mathrm{W}_{i,j,p,n} \cdot \mathrm{X}_{k+i-1,l+j-1,p} + b_k \quad , \qquad (1)$$

where i, j, p, n correspond to the dimensions of W. In this work, we apply different approximation and quantization levels

[1]These authors contributed equally to this work

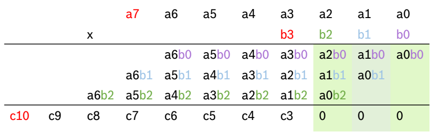
a7 a6 a5 a4 a3 a2 a1 a0
x b3 b2 b1 b0
a6b0 a5b0 a4b0 a3b0 a2b0 a1b0 a0b0
a6b1 a5b1 a4b1 a3b1 a2b1 a1b1 a0b1
a6b2 a5b2 a4b2 a3b2 a2b2 a1b2 a0b2
c10 c9 c8 c7 c6 c5 c4 c3 0 0 0

Fig. 1: Truncated multiplier with $T = 3$.

at each kernel, that is, at the dimension Ch_{out}, which in this case corresponds to the last dimension of weights and outputs.

In this work, we quantize 32-bit floating-point (FP) activations and weights to 8 and 4 bits respectively, following a linear quantization approach, characterized by a symmetric and uniform bin distribution. As this quantization results in accuracy degradation, the CNN is retrained until the FP accuracy is reached within a tolerance of 0.1%.

B. Truncated multiplier

Truncated multipliers are characterized by the truncation of T Least Significant Bits (LSBs) or *columns* in the partial product computation. We work with signed, 8×4 multipliers, affine to the utilized quantization. We denote the inputs as A (activation) and B (weight). In both operands, the Most Significant Bit (MSB) corresponds to the sign bit, as we use Sign-Magnitude (SM) representation. The multiplication is then performed between $a_0, ...a_6$ and $b_0, ..., b_2$ and finally we determine the resulting sign bit c_{10} through $a_7 \text{XOR} b_3$. The truncation scheme is presented in Fig. 1.

III. Hardware-aware CNN optimization

Our proposed methodology for hardware-aware CNN optimization is presented in Fig. 2. It consists of two stages: In the first stage, uniform CNN approximation is performed by iterative approximation and fine-tuning, and in the second stage, a further resiliency-based kernel-wise approximation is performed. Both stages are thoroughly described in the following sub-sections.

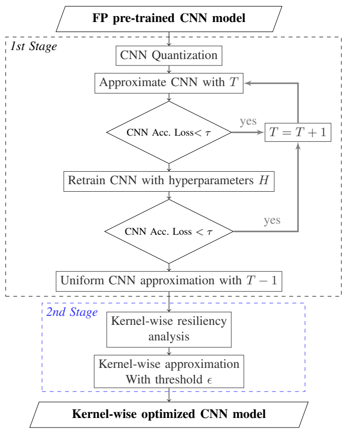
FP pre-trained CNN model
1st Stage
CNN Quantization
Approximate CNN with T
CNN Acc. Loss< τ
yes
$T = T + 1$
Retrain CNN with hyperparameters H
CNN Acc. Loss < τ
yes
Uniform CNN approximation with $T - 1$
2nd Stage
Kernel-wise resiliency analysis
Kernel-wise approximation With threshold ϵ
Kernel-wise optimized CNN model

Fig. 2: Methodology for hardware-aware CNN optimization

A. Full approximation and fine-tuning

In the first stage of our optimization approach, we quantize and then approximate all CNN layers using the truncated multiplier with $T = 1$. If the accuracy after approximation is within our proposed tolerance τ, we approximate the CNN with $T + 1$. If there is some accuracy degradation, the CNN is fine-tuned for few apochs until τ is reached. This process is repeated until τ cannot be reached after fine-tuning. With this, we guarantee a minimum approximation level, and thus energy savings, for the whole CNN.

B. Kernel-wise Optimization

After iterative full CNN approximation and fine-tuning, there is still room for further approximation at the kernel level, and therefore further energy savings. To achieve this, we propose to further approximate the most resilient kernels while maintaining our accuracy tolerance τ.

To select these resilient kernels, we propose a novel approach. This solution is based on understandable neural networks research [18], [19]. The focus of this research is to determine the relationships between neural network predictions and input feature maps. By linking the resulting prediction to the input stimulus through back-propagation, the most relevant input features can be identified. In this work, we make use of the framework presented in [19], where different methods to identify relevant input features in CNNs are implemented. This relevance is defined as the contribution of a certain pixel in the overall prediction of the network [18]. Hereby, we extend this concept to identify network elements where additional approximation can be applied with negligible CNN accuracy degradation. More specifically, we identify the least relevant kernels in each layer, and then quantize their weights to smaller bit-widths. This results in no additional approximation error, while achieving further power savings (see section V).

Note that to identify less resilient kernels, a threshold ϵ must be defined. This threshold is based on the average of the resiliency metric across the j and p dimensions and over the whole training set. To apply the kernel-wise optimization (KWO), Algorithm 1 is performed. The $ValidationAcc(M_{KWO,\epsilon})$ is computed according to the following approximation schemes: increase the value of T, or decrease the number of bits used to represent the kernel weights. Both approximations are evaluated in Section V.

The algorithm starts by performing a relevancy analysis using the whole training dataset. For each layer in the model, resiliency metrics are calculated through back-propagation between predictions and inputs. These metrics are then averaged over dimensions i, j, n (see (1)). These values are then normalized. A threshold $\epsilon = 1e{-}7$ is used in our experiments. We increase ϵ gradually by a factor α, as long as the test accuracy for the kernel-wise optimized model remains within the tolerance τ. Alongside LRP-presentAflat, we also test the following methods: Deep Taylor [7], LRP-epilson and LRP-Zplus [9], and GuidedBackprop [20]. Each method delivers different number of relevant kernels, and thus, different accuracy degradation, which is further analyzed in section V. Note that kernel relevancy and resiliency are inversely proportional.

IV. Hardware Implementation

The goal of the presented approximation techniques is to yield a more efficient hardware on both MAC unit level as well as on system level. In this section, we explore possible hardware realizations for both perspectives with focus on the first one, as the second is intended for future work.

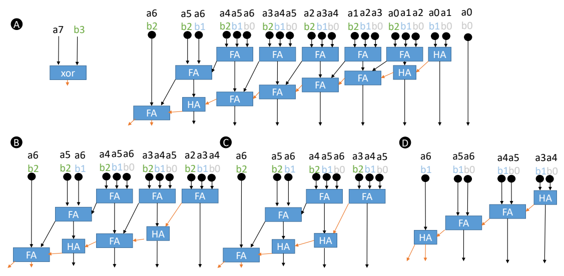

Fig. 3: Array Multipliers (A) A full 8*4 Array Multiplier including the sign computation, (B) A 4-column truncated 8*4 Array Multiplier, (C) A 5-column truncated 8*4 Array Multiplier, (D) A 4-column truncated 8*3 Multiplier

A. Approximated MAC Unit Level

The column truncation in the partial product of a multiplier can be translated into a MAC unit with less AND gates and adder blocks. In a CNN, as explained in section III, for all layers, and thus for all kernels, there is a minimum number of truncated columns that can be achieved. The truncated columns can then be completely discarded from the designed MAC unit, thereby saving power and area.

The second step is defining the columns that can be truncated or not depending on the kernel being computed. The hardware blocks intended for these column computations will exist in the final MAC unit. However, these blocks can be turned off completely, reducing the power consumption of the MAC unit. The main benefit would come from data-gated MAC units [21].

The blocks of truncated columns will be kept at zero. Therefore, switching activity is reduced. Moreover, as these blocks do not contribute anymore, the critical path of the MAC unit is drastically shortened. That could lead to either increase the operating frequency for faster throughput, or decrease the supply voltage at the same frequency and same throughput. In our approach we follow the second option which reduces the final power consumption. We illustrate such multiplier in Fig. 3, where we use an array-multiplier as the base design [22].

Algorithm 1 KWO Algorithm

Input: Model $\mathcal{M}$, Dataset $\mathcal{D}$
Output: Set of Output Kernel Indices I_k Satisfying the accuracy tolerance τ

```
1: Initialization $\epsilon = 1e^{-7}, \alpha = 10$
2: for all layers $l$ in $\mathcal{M}$ do
3:     $R_{i,j,p,n}$=LRP($\mathcal{M}^{(l)}, \mathcal{D}$)
4:     $R_n = \sum_i \sum_j \sum_p R_{i,j,p,n}$ {Compute average over data-sample, $j$ and $p$.}
5:     $R_n = R_n / \operatorname{argmax}_{n=1,\dots,N} \langle R_n \rangle$
6:     while $ValidationAcc(M_{KWO,\epsilon}) \geq \tau$ do
7:         $I_k$ = Find kernels such that $(R_n \leq \epsilon)$
8:         $\epsilon = \alpha\epsilon$
9:     end while
10: end for
```

B. System-level

The methodology we presented in section III can help to design more efficient system architectures. First, it can allow for systems with MAC units of different precision. For example, if the framework estimates that for all layers in a CNN at least a certain number of kernels can be approximated to a lower precision or a higher number of truncated columns, then the hardware can be built upon such information, consequently benefiting from it. Second, many research works target precision scalability from the layer perspective. In such architectures, according to the layer precision, further power saving and area efficiency can be achieved. This concept can be further extended to allow scalability within the same layer. For example, mapping kernels with the same approximation or truncation together to the same unit. As shown in Figure 3, one possible realization is having different MAC units with different approximations as part of the same system, and mapping the kernels to the appropriate MAC unit.

V. Experimental Results

We present the evaluation results of our proposed approach. We evaluate our methodology on three Residual Networks (ResNet) [16] trained for image classification with CIFAR10 [17]. In Table IV, we report the number of parameters and multiply operations of the evaluated CNNs, as well as the 8x4 quantized accuracies after retraining (see section II-A). All optimization experiments were performed using a GPU Nvidia GTX 1080 Ti with 11GB GDDR5X, and ProxSim, a GPU-accelerated framework for approximate computing [14], based on Tensorflow [23]. The different MAC units were compiled using Synopsys Design Compiler at 22nm FDSOI technology and supply voltage of 0.72v at original MAC and 0.59v at approximated version with same max frequency supported and worst operating conditions to model power and area of the synthesized components. The estimation of the power consumption was based on the number of MAC operations needed per layer, the power saving in each MAC through truncation, and the power saving in each MAC through switching from 4-bit quantized kernels to 3-bit quantized kernels.

A. Iterative approximation and fine-tuning

As a first step, a baseline is proposed in Table I. This baseline states the power saving for the first optimization stage, which results in uniform CNN approximation. In this first stage, performed according to Fig. 2, the following results were obtained for all ResNets:

- After 1 and 2 column truncation, the accuracy was still within $\tau = 1\%$ without fine-tuning.

TABLE I: Results - First Stage of Proposed Optimization Approach

Specs	ResNet 20		ResNet 14		ResNet 8	
	Trained Acc.	Power Saving	Trained Acc.	Power Saving	Trained Acc.	Power Saving
Baseline$^{(4,4)*}$	88%	28%	88%	28%	83%	28%
Upper Baseline I$^{(5,4)*}$	80%	38%	84%	38%	77%	38%
Upper Baseline II$^{(4,3)*}$	71%	40%	82%	40%	70%	40%

* (m,n) indicates m truncated columns and n bits quantized weights.

TABLE II: Results - Second Stage with Kernel-Wise Truncation $T = 5$

Resiliency Method	ResNet 20		ResNet 14		ResNet 8	
	Kernels Dist	Power Saving	Kernels Dist	Power Saving	Kernels Dist	Power Saving
LRP-PresentAflat	411/784	34%	201/560	32%	135/336	32%
Deep Taylor	250/784	31%	239/560	31%	120/336	30%
LRP-epislon	407/784	34%	244/560	32%	132/336	32%
LRP-Zplus	304/784	34%	176/560	30%	118/336	30%
Guided-Backpropagation	273/784	32%	172/560	32%	130/336	32%

TABLE III: Results - Second Stage with Kernel-Wise Weight Quantization to 3 bits

Resiliency Method	ResNet 20		ResNet 14		ResNet 8	
	Kernels Dist	Power Saving	Kernels Dist	Power Saving	Kernels Dist	Power Saving
LRP-PresentAflat	358/784	29%	201/560	28%	135/336	28%
Deep Taylor	250/784	26%	156/560	24%	107/336	25%
LRP-epislon	365/784	29%	207/560	28%	132/336	26%
LRP-Zplus	304/784	27%	176/560	25%	105/336	25%
Guided-Backpropagation	273/784	28%	172/560	27%	117/336	27%

TABLE IV: Parameters of ResNets

ResNet Instance	# conv. layers	# mults.	accuracy (floating-Point)	accuracy (qint-8)
ResNet-8	9	12.5M	85.68%	86.27%
ResNet-14	15	26.7M	89.41%	89.55%
ResNet-20	21	40.8M	91.04%	90.94%

- When truncating 3 and 4 columns, fine-tuning was performed over 10 epochs with a learning rate of 1e-4 to reach our accuracy tolerance.
- We find that at least 4 columns can be truncated from the MAC unit in all CNN layers while maintaining our proposed tolerance of 1% from the quantized accuracy.

In Table I, we report the accuracy obtained after this first optimization stage. We achieve 28% power saving in each network respectively. These values are estimated by using the saving in one MAC unit with 4 column truncation. in the whole CNN. For further comparison, we also propose two upper baselines for power savings: uniform approximation with $T = 5$, and uniform weight quantization to 3 bits.

B. Kernel-wise optimization

In Table II, the first results of our proposed kernel-wise optimization are reported. Here, we apply approximation by increasing the value of T to 5 in resilient kernels. The column *Kernels Dist* denotes the number of optimized kernels / total kernels. For each CNN, kernel-wise optimization is applied using all resiliency analysis methods introduced in section III. LRP-PresentAFlat and LRP-epsilon produce overall the largest number of resilient kernels to approximate with no additional error overhead. The results of these experiments highlight the advantages of optimizing kernel-wise, as even when truncating 4 columns, there is further room for optimization that was not achievable on the layer level. We achieve 34%, 32%, 32% energy savings for ResNet 20, 14, 8 respectively, with no accuracy drop nor further retraining, which is a significant improvement when compared to the upper baseline I.

In the last experiment, shown in Table III, the weights of the less relevant kernels are further quantized to 3 bits. Again, LRP based algorithms achieve better results, hence we conclude that LRP is the best method to obtain the resiliency of CNN kernels. Power savings of 29%, 28% and 28% were achieved respectively. We observe that further quantization of weights has a more severe effect on the accuracy than truncating additional columns. Therefore, the number of candidate kernels to optimize drops significantly, independently of the applied algorithm for resiliency analysis. This is also observed in the results reported in Table I, where truncating one additional column results in an accuracy drop of 8% in ResNet-20, while quantizing to 3-bit precision drops the accuracy by 17% in the same CNN. Similar results are reported regarding the other networks.

VI. Conclusion

In this paper, we introduce a novel approach to efficiently implement quantization and approximation in the computation of CNNs. By first applying uniform CNN approximation and then performing a kernel-wise resiliency analysis, we are able to combine low bit-width quantization and approximate multipliers to optimize computational resources of CNN applications without accuracy degradation. Additionally, we present possible hardware realizations of our proposed approach, at MAC unit and at system level. We validate our proposal through several experiments and demonstrate additional power savings of up to 34% in three different CNNs for image classification.

Acknowledgment

This work was funded by the ECSEL Joint Undertaking project TEMPO, in collaboration with the European Union's Horizon 2020 Research and Innovation Program and National Authorities, under grant agreement No. 826655.

References

[1] C. Gong *et al.*, “Mixed precision neural architecture search for energy efficient deep learning,” in *ICCAD*. IEEE, 2019, pp. 1–7.

[2] M. Masadeh *et al.*, “Input-conscious approximate multiply-accumulate (mac) unit for energy-efficiency,” *IEEE Access*, vol. 7, pp. 147 129–147 142, 2019.

[3] Y. Zhao *et al.*, “Automatic generation of multi-precision multi-arithmetic cnn accelerators for fpgas,” *ICFPT*, pp. 45–53, 2019.

[4] W. Choi *et al.*, “Sensitivity based error resilient techniques for energy efficient deep neural network accelerators,” *DAC*, pp. 1–6, 2019.

[5] D. Shin *et al.*, “Sensitivity-based error resilient techniques with heterogeneous multiply–accumulate unit for voltage scalable deep neural network accelerators,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, pp. 520–531, 2019.

[6] J. Zhang *et al.*, “Thundervolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep neural network accelerators,” *ArXiv*, vol. abs/1802.03806, 2018.

[7] G. Montavon *et al.*, “Explaining nonlinear classification decisions with deep taylor decomposition,” *ArXiv*, vol. abs/1512.02479, 2017.

[8] C. Schorn *et al.*, “An efficient bit-flip resilience optimization method for deep neural networks,” in *DATE*, 2019, pp. 1507–1512.

[9] G. Montavon *et al.*, “Methods for interpreting and understanding deep neural networks,” *Digit. Signal Process.*, vol. 73, pp. 1–15, 2018.

[10] S. Venkataramani *et al.*, “Axnn: Energy-efficient neuromorphic systems using approximate computing,” in *(ISLPED)*, Aug 2014, pp. 27–32.

[11] Q. Zhang *et al.*, “Approxann: An approximate computing framework for artificial neural network,” in *(DATE)*, 2015.

[12] C. Wu *et al.*, “Dynamic adaptation of approximate bit-width for cnns based on quantitative error resilience,” in *NANOARCH*, 2019.

[13] A. Marchisio *et al.*, “Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges,” in *ISVLSI*, 2019, pp. 553–559.

[14] C. D. L. Parra *et al.*, “Proxsim: Simulation framework for cross-layer approximate dnn optimization,” in *DATE*, March 2020.

[15] D. Esposito *et al.*, “Low-power approximate mac unit,” in *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, 2017, pp. 81–84.

[16] K. He *et al.*, “Deep residual learning for image recognition,” in *CVPR*, 2016.

[17] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *University of Toronto*, 2009.

[18] C. Schorn *et al.*, “Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators,” in *DATE*, 2018, pp. 979–984.

[19] M. Alber *et al.*, “innvestigate neural networks!” 2018.

[20] J. T. Springenberg *et al.*, “Striving for simplicity: The all convolutional net,” *CoRR*, vol. abs/1412.6806, 2015.

[21] B. Moons *et al.*, “Energy-efficient convnets through approximate computing,” in *WACV*, 2016, pp. 1–8.

[22] C. R. Baugh and B. A. Wooley, “A two’s complement parallel array multiplication algorithm,” *IEEE Transactions on Computers*, vol. C-22, no. 12, pp. 1045–1047, 1973.

[23] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. [Online]. Available: https://www.tensorflow.org/