Increasing Throughput of In-Memory DNN Accelerators by Flexible Layer-wise DNN Approximation

Cecilia De la Parra*, **Taha Soliman***, **Andre Guntoro**
Robert Bosch GmbH

Akash Kumar
TU Dresden

Norbert Wehn
TU Kaiserslautern

***Abstract*—Approximate Computing and Mixed Signal In-Memory accelerators are promising paradigms to significantly reduce computational requirements of DNN inference without accuracy loss. In this work, we present a novel in-memory design for layer-wise approximate computation at different approximation levels. A sensitivity-based high-dimensional search is performed to explore the optimal approximation level for each DNN layer. Our new methodology offers high flexibility and optimal trade-off between accuracy and throughput, which we demonstrate by an extensive evaluation on various DNN benchmarks for medium- and large-scale image classification with CIFAR10, CIFAR100 and ImageNet. With our novel approach, we reach an average of 5x and up to 8x speedup without accuracy loss.**

EMBEDDED INFERENCE OF DEEP NEURAL NETWORKS (DNNs) is highly challenging due to their large computational and energy efficiency requirements. However, DNNs are resilient to errors, and therefore paradigms such as approximate computing can significantly reduce their computational requirements while maintaining the accuracy. On the other hand, In-Memory Computing (IMC) offers a potential solution to the aforementioned requirements with increased throughput and energy efficiency.

Approximate computing can take several forms, from quantization [1], [2], to approximate hardware such as approximate multipliers, precision scaling [3] and approximate Processing Element (PE) design [4].

In many cases, selected hardware approximations are applied to all DNN neurons within the targeted network (*full approximation*). This approach is beneficial specially in accelerators where all features are computed with the same PEs [5]. More recently, full approximation of in-memory architectures has also been explored [4]. While significant energy savings can be reached, full approximation usually results in accuracy degradation that is not recoverable or requires further DNN optimization, for example *weight tuning* [6] or approximate DNN fine-tuning [5].

In contrast, *partial approximation* has been

*These authors contributed equally to this work

Published by the IEEE Computer Society

proposed [3] to maintain some computational savings, but without accuracy losses. In partial DNN approximation, sensitive DNN neurons/layers are computed with accurate PEs while resilient elements are computed with approximate ones [3].

Genetic search has also been used to find optimal configurations of approximate PEs for DNN computation [6]. However, these methods only allow for a limited number of approximate PEs, as the design space grows exponentially. For example, there is a total of 524287 possible configurations for VGG19 [7], in case of only choosing between a single approximate multiplier and an accurate one. Exploring such a design space relies on accuracy evaluation of each possible configuration to determine the optimal result, making the design space exploration highly prohibitive. Additionally, some of these methods also require re-training or weight-tuning, which significantly increases design space exploration required effort, even with advanced simulation frameworks for approximate computing [6], [5].

In this work, we present a novel DNN approximation approach that addresses the aforementioned challenges for DNN accelerators based on bit decomposition, especially in-memory architectures. We first quantize DNN activations and weights to 8 and 4 bit-INT respectively to minimize PE size and memory costs. Based on this data format, we approximate the product accumulation at bit-decomposed architectures, which allows for accurate or approximate computation at runtime without hardware modification/overhead. This additionally results in high flexibility, as accurate as well as approximated operations are computed with the same PEs. Inspired by the work in [2], we perform a layer-wise approximation sensitivity analysis with each possible configuration. This is followed by our proposed constrained graph search, which significantly reduces design space exploration efforts and delivers a Pareto set of accuracy vs. throughput. Summarizing, our new contributions are:

- Leveraging in-memory architecture design for flexible layer-wise accurate/approximate accumulation of bit-decomposed partial products.
- Layer-wise approximation sensitivity analysis of the proposed approximation scheme at different approximation levels.
- Multi-objective, sensitivity-based graph search for fast throughput-accuracy optimization.

Our new approximation scheme is evaluated using various DNNs for medium and large-scale image classification with CIFAR10, CIFAR100 [8] and ImageNet [9], where we are able to reach an average speedup of 5×, and of up to 8× with less than 1% accuracy loss.

IN-MEMORY ACCELERATION

Several in-memory accelerators have been recently introduced to leverage especially emerging memory technologies to perform multiply and accumulate (MAC) operations. However, the Analog-to-Digital Converters (ADC) used in these architectures usually represent a performance bottleneck, e.g. in ISAAC [10] the 8-bit ADC accounted for 58% of total power and 31% of the area. These requirements can be reduced by adopting different techniques at hardware-level such as partitioning each multiplication in smaller operands [11]. At bit level, such partitioning is referred to as *bit decomposition*. Many state-of-the-art architectures exploit bit-decomposition based algorithms to perform the MAC operation within the technology limitations such as single bit storage and to enable lower precision ADCs. The bit-decomposed MAC is computed by (1) and (2), where $f^{(t)}_{in_p,i}$ is the p bit of the input feature i at cycle t, w_{i_r} is the r-th bit of i-th DNN kernel, p corresponds to the bit-significance (0 corresponds to the LSB), q is the precision of w_i, k is the simultaneous possible activation and D is the total number of MAC operations per output feature f_{o_p}.

$$f^{(t)}_{o_p} = \sum_{r=0}^{q} \left(\sum_{i=0}^{k-1} f^{(t)}_{in_p,i} * w_{i_r} \right) << q \quad , \tag{1}$$

$$f_o = \sum_{t=0}^{D/k} \sum_{p=0}^{P} f^{(t)}_{o_p} << p \quad , \tag{2}$$

In result, several in-memory architectures perform the MAC operation using the structure shown in Figure 1 where the DNN weights are stored across the memory cells and the activations are applied serially to stored weights to perform AND operations. The results are accumulated and further processed using adders and shifters to yield the final output feature. These in-memory

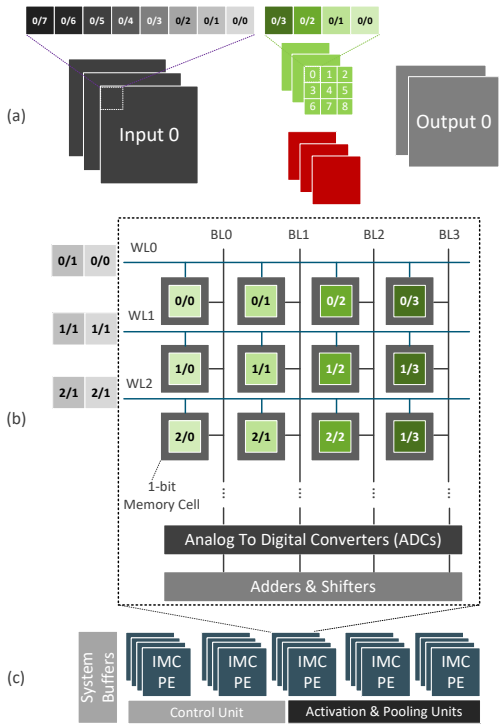


Figure 1: (a)Input and kernel words breakdown for convolution. (b)IMC Processing Element (PE) design where weight line applies the input signal while the bit line accumulates the result which is transferred to digital domain using ADC to be further processed. (c)Typical In-memory architecture including PEs enough to accommodate target networks as well as buffers, processing blocks and control unit.

architectures leverage such structure to enable more efficient ADCs with lower precision range. This has been explored with technologies such as FeFETs[12] and SRAM[13].

Proposed in-memory approximation

In non-approximate in-memory computing architectures, the precision of the connected ADCs to the crossbar must match the maximum number of activated memory cells per clock cycle. However, due to the resiliency of DNNs, it is possible in some DNN layers to activate more memory cells than the ADC precision which leads to an approximate result in case the accumulated value is larger than the ADC precision. Since ADCs usually constitute a major part of area and power consumption of the in-memory architecture, such approximation has a direct influence on the architecture efficiency. Additionally, this approximation allows for exploitation of the bit-level sparsity that can be offered by the decomposition of the MAC operation. This approximation is not limited only to mixed-signal in-memory computing architectures but also applies to a wide range of accelerators. For example, in digital based acceleration such approximation can drastically reduce the adders and accumulator size. We focus in this paper on applying our proposed approximation to the mixed-signal in-memory architectures as they show the highest potential in terms of efficiency gains. As shown in (3), the approximation can be applied by limiting the accumulated bits sum to the precision of the ADC represented by m.

$$\widehat{f}_{o_p}^{(t)} = \sum_{r=0}^{q} \min\left(\sum_{i=0}^{k-1} f_{in_p,i}^{(t)} * w_{i_r}, m\right) << q \quad , \tag{3}$$

$$\widehat{f}_o = \sum_{t=0}^{D/k} \sum_{p=0}^{P} \left(f_{o_p}^{(t)} + \epsilon_{approx}\right) << p \tag{4}$$

The approximation error ϵ_{approx} yielded by (3) is then:

$$\epsilon_{approx} = \begin{cases} 0 & \text{if } \sum_{i=0}^{k-1} f_{in_p,i}^{(t)} * w_{i_r} \leq m \\ \sum_{i=0}^{k-1} f_{in_p,i}^{(t)} * w_{i_r} - m & \text{otherwise} \end{cases} \tag{5}$$

For efficient approximation, we propose to increment the parallel possible activations k by multiples of 2, while maintaining the ADC precision m. The used approximation, denoted by the value k, is applied layer-wise. To compute a DNN, we have a possible set of configurations $\mathcal{K} = [k_1, k_2, ..., k_c]$, E.g. with a 3-bit ADC, we can have the set [8, 12, 16, 20, 24] ($c = 5$) where $k_1 = 8$ corresponds to the exact computation, and $k_c = 24$ is the maximum approximation level with a possible speedup up to 3x. This approach effectively increases throughput without any additional modifications in hardware.

A major challenge is to find the optimal value of k for each layer that can maximize the speedup without compromising the network accuracy. A further challenge is to apply such approximation on less resilient quantized networks. In the

following Section, we explain the quantization method applied to the DNN benchmarks in order to reduce computational requirements before applying any approximation. Then, we present our proposed method to determine the optimal value of $k \in \mathcal{K}$ for each DNN layer.

LAYER-WISE OPTIMIZATION

Quantization

To significantly increase the throughput and energy efficiency as well as insuring the applicability of our approximations to optimized hardware implementations, we perform linear and symmetric quantization computed by (6) [1], where B is the number of bits and Δ_z the *quantization step*. We quantize DNN parameters to 4 bits and activations to 8 bits in INT format, and consequently refer to this as 8A4W quantization format. Quantization steps Δ_z are optimized by minimizing the propagated quantization error [1]. The quantized DNN is then re-trained to maintain the proposed accuracy tolerance.

$$z_q = \text{clip}\left[\text{round}\left(\frac{z}{\Delta_z}\right), -2^{B-1}, 2^{B-1}-1\right]\Delta_z \quad (6)$$

This quantization approach can be scaled to larger bit-widths for both weights and activations. The use of more bits (e.g. 8-bit weights) will result in reduced accuracy loss, and therefore quantized re-training might not be necessary.

Methodology

The approximation of a DNN can be optimized by minimizing: 1) the loss ϕ between the outputs of the approximated model $\widetilde{f}(\mathbf{K}, x)$ and training samples y, and 2) the execution times $t(\widetilde{f}(\mathbf{K}))$ of the approximated model:

$$\min_{\mathbf{K}}\left(\frac{1}{N}\sum_{i\in\mathcal{N}}\phi(\widetilde{f}(\mathbf{K}, x_i), y_i), t(\widetilde{f}(\mathbf{K}))\right), \quad (7)$$

where $\mathbf{K}$ is the set of configurations used for layer-wise approximation of the target DNN, x_i, y_i represent the inputs and labels of training sample i respectively, of $\mathcal{N}$ total samples. For classification tasks, the DNN outputs y and $\widetilde{f}(\mathbf{K}, x_i)$ are probability distributions.

To efficiently minimize (7) we propose the methodology shown in Alg. 1. This methodology

Algorithm 1 Sensitivity-based DNN approximation

Require: DNN model $f(x)$ quantized to A_8W_4, training mini-batch x, max. throughput t_{max}
Output: Pareto front of approximate configurations P

for layer l in $f(x)$ and k in $\mathcal{K}$ **do**
  Compute $\mathbf{KL}_l(k)$
end for
Generate GRAPH with single NODE η_0
for k in $\mathcal{K}$ **do**
  for NODE j in GRAPH **do**
    NODES$_{new}$ $\leftarrow$ []
    for l in $f(x)$ **do**
      Generate NODE η_{new}:
      η_{new}.cost = η_j.cost + $t_l(k)$
      η_{new}.sens = η_j.sens + $\mathbf{KL}_l(k)$
      Append η_{new} to NODES$_{new}$
    end for
  end for
  GRAPH $\leftarrow$ Pareto-front(NODES$_{new}$)
end for
return $P \leftarrow$ GRAPH

is designed to find an optimal trade-off between throughput and accuracy, depending on the system's requirements. By applying our flexible approximation concept, throughput can be increased by computing less sensitive layers with more aggressive approximations, while sensitive layers are computed more accurately. The challenge is to find an optimal configuration $\mathbf{K} = k_1, ..., k_L$ where $k_l \in \mathcal{K}$ is the max. number of parallel possible activations k for the computation of layer l without compromising the model accuracy. Although the ADC resolution m is fixed, the search space grows exponentially with each possible configuration in $\mathcal{K}$, which results in a highly prohibitive design space exploration. For example, in case of approximating VGG19 [7] with 5 different configurations that results in 19 trillion possible configuration. To explore such design spaces, we instead measure the sensitivity of the 8A4W quantized DNN to the approximation of each layer with each different configuration. This sensitivity measurement is used to estimate the cumulative sensitivity, which is proportional to the DNN accuracy degradation. Accuracy evalu-

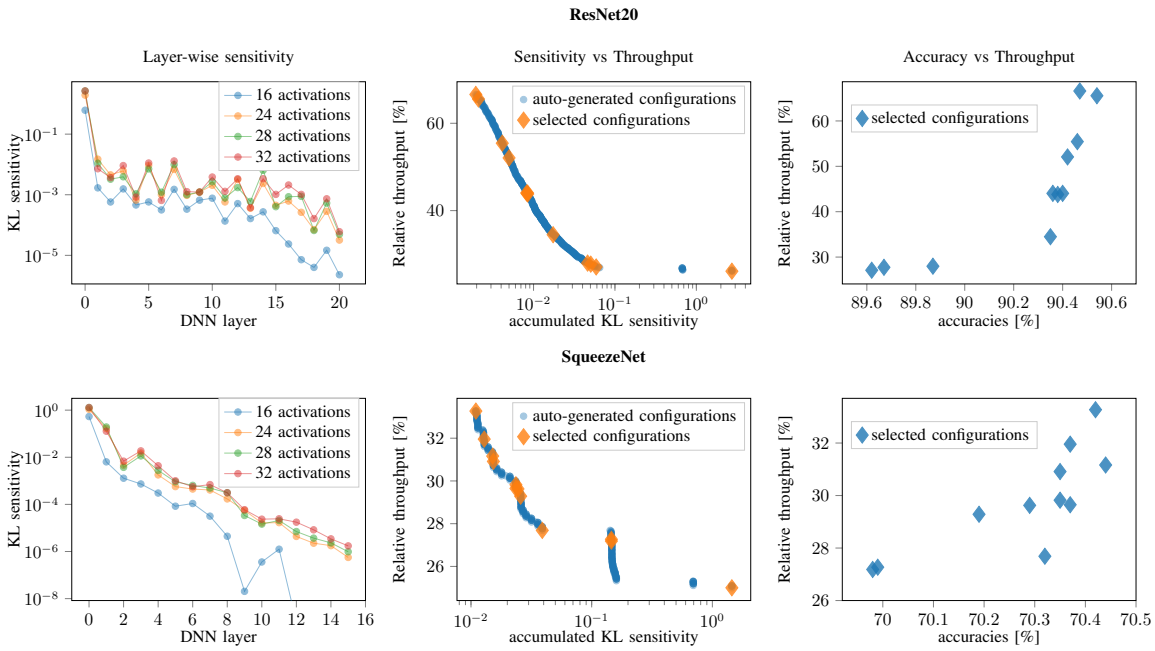


Figure 2: Sensitivity, accuracy and throughput of generated approximate configurations for ResNet20 and SqueezeNet, given the following configuration: [8, 16, 24, 28, 32]. The first column presents the sensitivity of each DNN layer computed with each given configuration. In the second column, accumulated DNN sensitivity vs. relative throughput of all generated solutions is plotted. In the third column, the DNN accuracy of a small selection of the generated solutions is evaluated.

ation substantially increases design space exploration experiments time, as this requires the DNN evaluation over the complete dataset. Inspired by the work in [2], we instead leverage the Kullback Leibler (KL) divergence as sensitivity metric. The KL divergence is defined by (8), where $f(x_{i,q})$ is the output of the quantized DNN model without approximation, and $\widetilde{f}(k_l; x_{i,q})_k$ is the output of the quantized model with l-th layer approximated with k. The advantages of KL divergence against other metrics, such as accuracy, are: 1) It is more adequate to measure the difference between probability distributions, such as the outputs of the DNN models evaluated in this work. 2) To obtain (8) we only require a mini-batch of training samples (e.g. only 40 samples), which significantly reduces space exploration efforts.

$$\mathbf{KL}_l(k) = \sum_{i \in \mathcal{N}} f(x_{i,q}) \log \frac{f(x_{i,q})}{\widetilde{f}(k_l; x_{i,q})_k} \qquad (8)$$

Once the sensitivity of each layer computed with each possible approximation $k \in \mathcal{K}$ is obtained, we perform a graph search according to Alg. 1 to solve the optimization problem in (7). Here, a graph is defined as a set of nodes, each one representing a possible approximate configuration $k \in \mathcal{K}$ for each DNN layer l. At each iteration: 1) the graph is extended forward by adding a new graph node. This node corresponds to the current DNN layer and contains all possible approximate configurations for said layer. 2) A Pareto front of sensitivity vs. rel.throughput of the existing nodes within the graph is generated. Consequently, we prune inefficient nodes early on, substantially reducing the design space.

The use of KL divergence as sensitivity metric does not guarantee that the solution is the actual Pareto-front of accuracy and throughput. However, our experiments demonstrate that the KL divergence is almost proportional to accuracy and therefore a good metric to generate the set of optimized solutions.

EVALUATION

In this Section, we present a) our experimental setup and simulation details, and b) the results obtained with our proposed flexible approximation

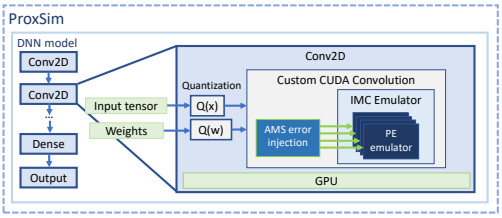


Figure 3: IMC architecture simulation in ProxSim, where Q(.) represents tensor quantization. The AMS error injection is performed at PE level. The custom CUDA operator allows for GPU acceleration.

methodology.

Setup

The approximate computing framework ProxSim [5] was adapted for a bit-accurate simulation of the proposed approximations. For this, a custom CUDA operator was implemented to simulate the bit-decomposed IMC architecture from Figure 1. As reported in [11], the Analog-Mixed Signal (AMS) must be taken into account for reliable IMC simulation. Therefore, we include the AMS error from the target architecture in our experimental setup. The simulation is performed as illustrated in Figure 3. In case of the architecture from [12] (3-bit ADC), the AMS error is obtained from the FeFET variability, using Monte Carlo simulations of a single PE and confirmed with taped-out wafer measurements, considering all possible inputs and outputs (both Gaussian-distributed). This noise model is injected at PE level to the custom CUDA operator.

In [12], the 3-bit ADC accumulation noise does not have any impact after digitalization, and therefore is not included in the AMS noise model. For the simulation of the 4-bit ADC architecture, we assume that the non-linearity model from [13] is considered during DNN training. The hardware performance estimations were obtained from the cell activation power as well as the architecture performances published in [12], [13].

Results

We evaluate various DNNs for large-scale image classification with CIFAR10/100 and ImageNet: ResNet [14], Network-in-Network [15], VGG19 [7] and SqueezeNet [9]. Their Floating-point (FP) accuracies as well as the 8A4W quantized accuracies before and after fine-tuning (FT) are presented in Figure 4a). We observe that the performed fine-tuning successfully compensates the quantization error with negligible accuracy losses. The 8A4W accuracy is consequently considered as baseline for the evaluation of the proposed approximation technique. We evaluate the impact of several per-layer combinations of accumulated activations, presented in Figure 4b). With this, we aim to demonstrate that our approach is not only applicable to our target in-memory architecture with 3-bit ADC [12], but can also be applied to 4-bit ADC in-memory architectures such as [13]. The performance of the target 3-bit ADC macro based on [12] is presented in Figure 4f).

To illustrate the optimization process from Alg. 1, we present in Figure 2 the sensitivity, accuracy and throughput values obtained when approximating ResNet20 and SqueezeNet with $\mathcal{K} = [8, 16, 24, 28, 32]$. In the first plot column, the layer-wise sensitivity computed by (8) is presented. For both plotted layers, as well as in the rest of the evaluated DNNs in this work, the first layer is by far the most sensitive and therefore is computed accurately or at limited approximation configuration. Additionally, we observe that for large-error approximations (in this example of 24 or more activation accumulation), the sensitivity is similar for most of the layers, which allows to apply more drastic approximations without significant accuracy loss. In the second plot column, we present all auto-generated solutions using Alg. 1, which build a pseudo-Pareto front of relative throughput and accumulated sensitivity of all possible configurations within the design space. We select a representative number of all auto-generated solutions to evaluate the final DNN accuracy after approximation. As expected, the accumulated DNN sensitivity is proportional to the accuracy degradation due to our proposed approximations, and therefore the solution found by Alg. 1 delivers an optimal accuracy-throughput trade-off. When the AMS error model is included in the computation, the final accuracy has a max. deviation of $\pm 0.1\%$. An example of a found configuration by our proposed methodology is

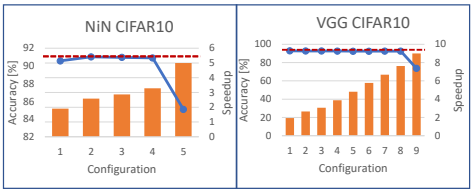
a) Evaluated DNNs

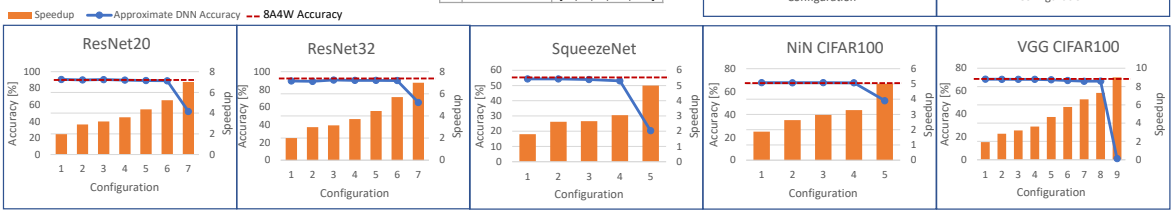
DNN	Dataset	#MAC ops.	FP Accuracy	8A4W acc. Before FT	8A4W acc. After FT
ResNet20	CIFAR10	40.8	91.14	78.54	**90.53**
ResNet32		69.1	91.92	61.71	**91.11**
NiN		222.5	90.75	88.87	**91.04**
VGG19		417.1	93.22	84.07	**92.78**
NiN	CIFAR100	223.5	67.68	52.85	**66.59**
VGG19		400.6	72.57	38.06	**70.57**
SqueezeNet	ImageNet	349.5	56.67	24.86	**54.20**

b) Evaluated approx. Configurations

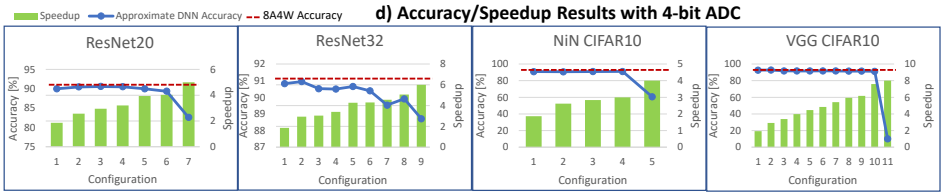
	Approx. Configuration activations	
	3-bit ADC	4-bit ADC
1	[8,12,**16**]	[16,24,**32**]
2	[8,12,16,20,**24**]	[16,24,32,40,**48**]
3	[8,16,20,24,**28**]	[16,32,40,48,**56**]
4	[8,16,24,28,**32**]	[16,32,48,56,**64**]
5	[8,16,24,32,**40**]	[16,40,48,64,**72**]
6	[8,24,32,40,**48**]	[16,40,64,72,**80**]
7	[8,32,40,48,**56**]	[16,40,72,80,**88**]
8	[8,40,48,56,**64**]	[16,40,72,88,**96**]
9	[8,40,56,64,**72**]	[16,40,88,96,**104**]
10	---	[16,40,96,112,**128**]
11	---	[16,40,96,128,**144**]

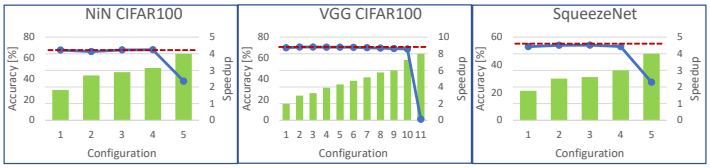
c) Accuracy/Speedup Results with 3-bit ADC





d) Accuracy/Speedup Results with 4-bit ADC





e) Example of found config.

Layers	VGG19 – CIFAR10 Activation configuration	
	3-bit ADC [8,12,16]	4-bit ADC [16,24,32]
1st	8	16
2nd	16	24
rest	16	32

f) 3-bit ADC Architecture Performance

Activations	TOPS/W
8	46.9
12	66.9
16	85.1
20	101.7
24	116.8
28	130.8
32	143.6
40	166.5
48	186.7
56	203.6
64	218.9
72	232.4

Figure 4: a) Evaluated DNNs num. of parameters, MAC ops and accuracies at Floating point(FP) representation as well as at 8A4W before and after fine-tuning (FT). b) All evaluated approximate configurations for 3-bit and 4-bit ADCs. c) 3-bit ADC solutions with best accuracy/speedup trade-off for each configuration in b) that maintains the 8A4W DNN accuracy. d) 4-bit ADC solutions with best accuracy/speedup trade-off for each configuration in b) that maintains the 8A4W DNN accuracy. e) Example of found configurations for model VGG19 for CIFAR10 (3-bit and 4-bit architecture, 1st set of possible configurations): 1st and 2nd layers require more accuracy, rest of layers can be computed with max. approximation level. f)Performance of 3-bit ADC in-memory accelerator from [12] with each possible activation configuration.

provided in Figure 4e).

For each configuration given in Figure 4b), we obtain the optimal configurations by applying Alg. 1. In Figure 4c) and 4d), we present the accuracy and throughput of the solutions that provided the best accuracy-speedup trade-off with each given configuration, while maintaining the original accuracy. Additionally, the last bar in each chart represents the next possible speedup and its corresponding accuracy drop. The highest speedup is reached with VGG19 (7.6x for CIFAR10 and 7.3x for CIFAR100). The reason for this is: The VGG19 has a large number of parameters and MAC operations compared to the other evaluated models. This results in a high redundancy of convolutional features that contribute to the accuracy of the DNN output. In consequence, this increases the model's robustness towards errors in the DNN computation, including approximation errors. For the rest of the evaluated models, we are able to reach an average of 5x speedup without accuracy loss.

CONCLUSION

In this work, we present a novel layer-wise approximate computing technique targeting in-

memory architectures as well as accelerators based on bit decomposition of MAC operation. Using a sensitivity-based search, we identify the optimal approximation level for each DNN layer. We performed several experiments with a wide variety of approximate configurations to validate our approach at various DNN benchmarks for image classification. With the obtained results, we reached an average speedup of 5x without accuracy loss and without any change to the accelerator or the architecture itself, demonstrating the suitability of our proposed approximation approach for accuracy-throughput trade-off optimization.

ACKNOWLEDGMENT

This work was funded by the ECSEL project TEMPO, the European Union's Horizon 2020 Research and Innovation Program, and National Authorities, under grant agreement No. 826655 and partially funded by Carl Zeiss foundation under grant "Sustainable Embedded AI".

REFERENCES

1. S. Vogel, J. Springer, A. Guntoro, and G. Ascheid, "Self-supervised quantization of pre-trained neural networks for multiplierless acceleration," in *Des.Aut.Test.Eu.Conf.Ex.*, 2019, pp. 1094–1099.
2. Y. Cai *et al.*, "Zeroq: A novel zero shot quantization framework," in *Conf.Comp.Vis.Pattern Recogn.*, 2020.
3. Q. Zhang *et al.*, "ApproxANN: An approximate computing framework for artificial neural network," in *Des.Aut.Test.Eu.Conf.Ex.*, 2015, pp. 701–706.
4. T. Soliman *et al.*, "Adaptable approximation based on bit decomposition for deep neural network accelerators," in *Int.Conf.Art.Intell.Circ.Sys.*, 2021, pp. 1–4.
5. C. De la Parra *et al.*, "Proxsim: Gpu-based simulation framework for cross-layer approximate dnn optimization," in *Des.Aut.Test.Eu.Conf.Ex.*, 2020, pp. 1193–1198.
6. V. Mrazek *et al.*, "ALWANN: automatic layer-wise approximation of deep neural network accelerators without retraining," in *Int. Conf. Comp. Aided Des.*, 2019.
7. K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Repr.*, 2015.
8. A. Krizhevsky *et al.*, "Cifar-10," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html
9. F. N. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *Proc. Int. Conf. Learn. Repr.*, 2016.
10. A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd An.Int.Symp. on Comp.Arch. (ISCA)*, 2016, pp. 14–26.
11. A. S. Rekhi, B. Zimmer *et al.*, "Analog/mixed-signal hardware error modeling for deep learning inference," in *2019 56th ACM/IEEE Des.Aut.Conf. (DAC)*, 2019, pp. 1–6.
12. T. Soliman *et al.*, "Ultra-low power flexible precision fefet based analog in-memory computing," in *Int. Elect. Dev. Meet.*, 2020, pp. 29.2.1–29.2.4.
13. Q. Dong *et al.*, "15.3 a 351tops/w and 372.4gops compute-in-memory sram macro in 7nm finfet cmos for machine-learning applications," in *Int. Solid- State Circ. Conf.*, 2020, pp. 242–244.
14. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conf.Comp.Vis.Pattern Recogn.*, 2015.
15. M. Lin *et al.*, "Network in network," in *Proc. Int. Conf. Learn. Repr.*, 2014.

Cecilia De la Parra is a Ph.D. candidate at Robert Bosch GmbH, in collaboration with the Chair of Processor Design at the TU Dresden. Her primary areas of research interest include digital approximate computing and optimization of cross-layer approximations in DNNs. Contact her at cecilia.delaparra@de.bosch.com

Taha Soliman (S'20) is a Ph.D. candidate at Robert Bosch GmbH, in collaboration with the Chair of Microelectronic Systems Design at the TU Kaiserslautern. His primary areas of research interest include in-memory computing, emerging technologies and approximate computing for DNNs. Contact him at taha.soliman@de.bosch.com

Andre Guntoro (M) received in 2009 his Dr.-Ing. Degree in Microelectronics from TU Darmstadt with focus on flexible wavelet processor design and implementation. Since 2011 he joined Robert Bosch GmbH in Research Division, focusing on embedded Machine Learning, where he leads the research activities in Machine Learning and Deep Learning specialized for automotive and IoT devices.

Akash Kumar (SM'13) received the joint Ph.D. degree in electrical engineering and embedded systems from the Eindhoven University of Technology, and the National University of Singapore (NUS), Singapore,

in 2009. From 2009 to 2015, he was with NUS. He is currently a Professor with TU Dresden, Germany, where he is directing the Chair for Processor Design. His current research interests include the design, analysis, and resource management of low-power and fault-tolerant embedded multiprocessor systems.

Norbert Wehn (SM) received the Diploma and Ph.D. degrees from the TU Darmstadt, Germany. He currently holds the Chair for Microelectronic Systems Design at the Department of Electrical and Computer Engineering, University of Kaiserslautern, Germany. His special research interests are VLSI architecture for mobile communication, forward error correction techniques, low-power techniques, advanced SoC and memory architectures, 3-D integration, reliability issues in SoC, the IoT, and hardware accelerators for big data applications.