Prediction of the Impact of Approximate Computing on Spiking Neural Networks via Interval Arithmetic

Prediction of the Impact of Approximate Computing on Spiking Neural Networks via Interval Arithmetic

Sepide Saeedi[1], Alessio Carpegna[1], Alessandro Savino[1], and Stefano Di Carlo[1]
[1]*Control and Computer Eng. Dep., Politecnico di Torino* Torino, Italy
{sepide.saeedi, alessio.carpegna, alessandro.savino, stefano.dicarlo}@polito.it

***Abstract*—Approximate Computing (AxC) techniques allow trade-off accuracy for performance, energy, and area reduction gains. One of the applications suitable for using AxC techniques are the Spiking Neural Networks (SNNs). SNNs are the new frontier for artificial intelligence since they allow for a more reliable hardware design. Unfortunately, this design requires some area minimization strategies when the target hardware reaches the edge of computing. In this work, we first extract the computation flow of an SNN, then employ Interval Arithmetic (IA) to model the propagation of the approximation error. This enables a quick evaluation of the impact of approximation. Experimental results confirm the model's adherence and the capability of reducing the exploration time.**

***Index Terms*—approximate computing, spiking neural networks, interval arithmetic**

I. Introduction

Approximate Computing (AxC) techniques trad-off the computation accuracy for performance gains, e.g., reductions in power consumption, memory utilization, and execution time [1]. Hence, applications intrinsically tolerant to some loss in computation accuracy, like Artificial Neural Networks (ANNs), are good targets for employing AxC techniques [2]. Running ANNs demands executing billions of arithmetic operations and storing millions of weights. So, applying AxC techniques to ANNs seems promising in terms of gains in performance, targeting a low-power device [3]–[5].

What makes choosing among different AxC techniques difficult is that all of them can be employed together, requiring the evaluation of the impact of each combination of AxC techniques on the overall computation accuracy. The two main approaches for evaluating this impact, comprise running the application several times with different configurations [6], [7] or devising modeling techniques to consider them in a time-optimized fashion [8], [9]. The first approach's disadvantage is that the evaluation time increases when the exploration reaches an exhaustive search. Applying pruning techniques to the exploration space, reduces the exploration time to the application's execution time at best. Conversely, the second approach is more suitable for estimating the impact of AxC techniques on the application computation accuracy because it is faster than the first approach, even though it costs an error margin in the computation accuracy evaluation. Hence, we chose the modeling approach over the first approach since it gives a quicker estimation.

Spiking Neural Networks (SNNs) are Artificial Neural Networks that, like Deep Neural Networks (DNNs), can benefit from employing AxC techniques [10]. In SNNs, the information is exchanged between neurons as binary spikes [11]. While SNNs have shown a great potential for achieving high accuracy, they require a small area footprint, thus potentially limiting the power/energy consumption due to their sparse spike-based operations [12]. Moreover, SNNs support unsupervised learning with unlabeled data using the Spike-Timing-Dependent Plasticity (STDP) [13]. Each neuron's computation flow in an SNN comprises arithmetic operations and necessary weights and thresholds to operate. Therefore, when deploying SNNs into FPGA devices, employing AxC techniques can help decrease the storage required for weights and thresholds and the complexity of the arithmetic components.

It is noteworthy that, in this paper, we refer to two different terms using the word accuracy: computation accuracy and network accuracy. Computation accuracy refers to the extent to which the result of an arithmetic operation is accurate, providing that the operator or the operand(s) of the arithmetic operation are approximated. The same definition applies to a complex computation comprising several arithmetic operations. While network accuracy refers to the extent to which the neural network result is accurate. For example, if the neural network is performing an image classification task, the probability of the event that the network can classify the image in the right class is defined as its accuracy.

In this work, we resorted to Interval Arithmetic (IA) [14] concepts to explore the impact of employing AxC techniques on spiking neural networks. The exploration aimed to evaluate the effect of data reduction approximation techniques in fixed-point quantization of all weights and thresholds. Some previous works [15], [16] exploit interval arithmetic concepts to analyze neural network computations in general. Nevertheless, in these works, every single value propagating during the calculation does not allow to distinguish the introduced error. On the contrary, we aimed to define a representation to report the approximation-induced error range and provide further tuning optimization opportunities. In fact, unlike previous works [15], [16], we exploit interval arithmetic concepts to model the errors introduced by approximation to each network parameter and propagate it through network computations to observe the impact of approximation-induced errors on the final computation accuracy. Then, we conducted experiments on a trained spiking neural network [17], exploring the parameters' size reduction to find the most significant reduction to let the SNN better fit into the FPGA. To evaluate the quality of

the model, the outputs of all approximated versions of the SNN were compared to those obtained from the IA-based model.

Our goal was the design space exploration, and our approach can help find the best reduction of the data precision such that the network accuracy is still acceptable, before FPGA-based design. If the FPGA-based design is flexible and allows to set the data precision, analyzing the impact of the precision reduction techniques requires changing the design, and re-synthesizing, for each reduction of a specific number of bits. So, it would be more time-consuming to study the impact of AxC techniques on the computation accuracy by directly performing inference on the FPGA. Furthermore, the proposed IA model can be generalized for other applications with complex computations, for example, NN designs, as long as we can model the network computation flow. For instance, our model can be generalized to help in accelerator design for convolutional SNNs, like those in previous works [18], [19].

The rest of this paper is organized as follows: section II describes the methodology, including an overview of a single neuron and the SNN architecture. The experimental results are provided and analyzed in section section III. Eventually, section IV draws conclusions and possible future works.

II. Methods

In this section, first, the spiking neural network is described, followed by a description of the data quantization and precision reduction and how it was applied to the trained SNN. Finally, the IA modeling is explained.

A. Spike Neuron Model and Network Description

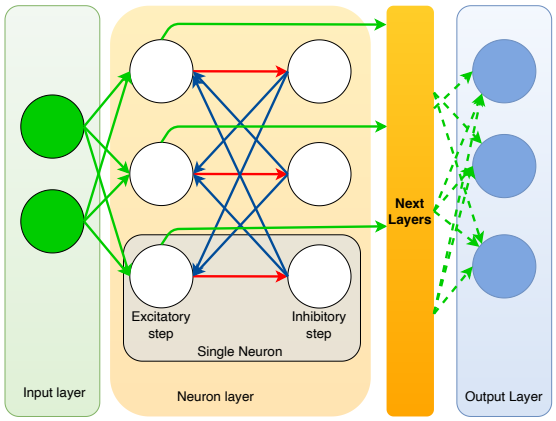

Fig. 1: An excitatory and an inhibitory layer of an SNN

Figure 1 depicts the general organization of an SNN, showing the behavior of each single spiking neuron concerning the others in the layer. Each neuron is composed of two computational steps: the excitatory step and the inhibitory step. Spikes are generated by an input layer transforming data, i.e., pixels of an image, into a sequence of spikes. Each spike is represented as a single bit set to 1 and enters the neuron through the excitatory step (green arrows from the input layer). This step is responsible for accumulating a value, the *membrane potential* (V_0 in Figure 2), based on the spikes at its inputs. It also generates an output spike when that value reaches a specific threshold (the green arrows out the excitatory step). Those spikes serve as input for the inhibitory step (the red arrow) and any other neurons' layer connected to it (the connection to the next layers). The inhibitory step models the interaction with all other neurons in the layer, and it acts by decreasing the neuron's *membrane potential* when they generate spikes. Eventually, the final layer is connected to an output layer that receives spikes and transforms them into useful information to assess the network's response. To appreciate the computation flow more clearly, Figure 2 shows the organization of a single layer.

The increment of V_0 is based on a weight value elaborated during the training phase and associated with each connection to the neuron ($w_{0,i}$ for the first neuron in Figure 2). Each time spikes enter the neuron, V_0 increases by the sum of all weights where active spikes are detected in the input connections. This can be generalized as shown in Equation 1. Where n is the number of input connections of the neuron, and inp_i is a binary value that shows the presence of an active spike when set to 1, otherwise 0. Here, $w_{j,i}$ shows the value of each weight for neuron j (as it happens for the 0 one in Figure 2).

$$sum_j = \sum_{i=0}^{n-1} inp_i \cdot w_{j,i} \tag{1}$$

The *membrane potential* (V_0 in Figure 2) behaves like an "electrical charge.": when it reaches its maximum ($V_0 > V_{thresh}$ in Figure 2), the neuron fires a spike and it is reset to a special *reset* value (V_{reset} as it can be seen following the "yes" branch in Figure 2). V_{thresh} is computed at training time. The *membrane potential* falls at each step if no active spike is detected in the input using an exponential decay approach ($V_0 \gg exp_{decay}$ in Figure 2). The decay is a far more complex operation involving multiplication for an exponential time-related term, but, to occupy less area on an FPGA, the work in [17] proposes a transformation of the exponential term into the closest power of two to replace the multiplication with a shift. Since the exponent is negative, the actual shift is a right shift.

Finally, in [17], each neuron is associated with a counter to keep track of the output spikes generated by its excitatory layer and define the output of the SNN. The presence of counters is not mandatory and is particular to the SNN exploited in this work. For more detailed information about the model and implementation, the reader may refer to [17].

B. Data Quantization and precision reduction

The weights, thresholds, and reset values are computed during the training phase. Since most training tools available calculate these values in a floating-point fashion [20], an extra step is required to deliver the network to edge devices provided by some FPGA accelerators. To do so, in [17] all values were quantized to reduce the memory occupation.

The quantization technique requires converting each floating-point value into a 32-bit wise fixed-point value, with the fractional part of 16 bits, as in Equation 2.

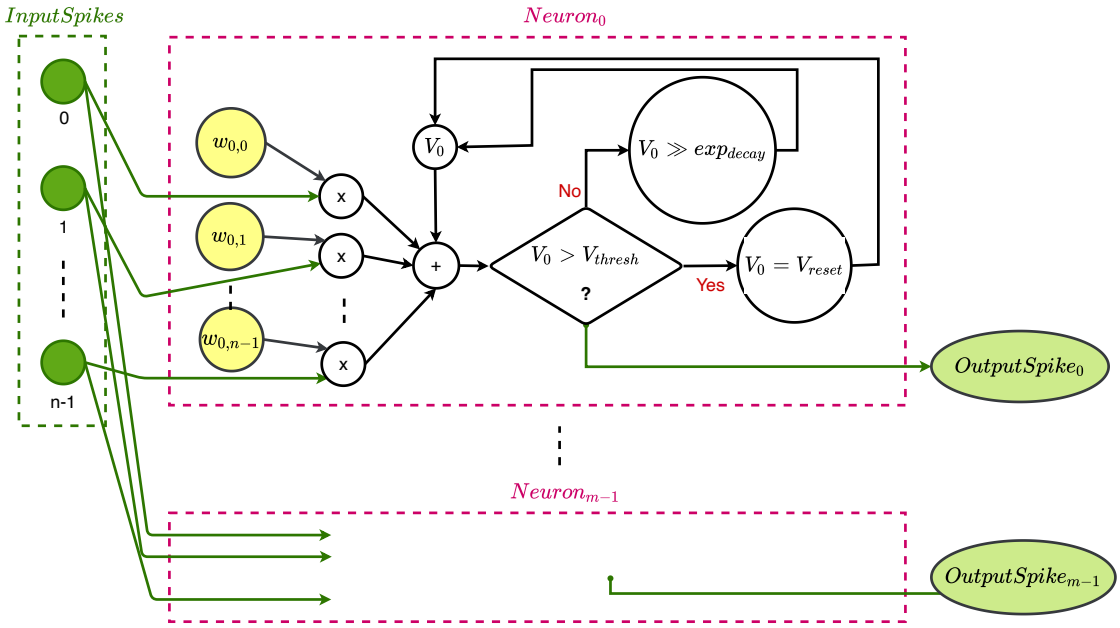

Fig. 2: Computation Flow of an SNN layer

$$v = \sum_{i=16}^{31} b_i \cdot 2^{(i-16)} + \sum_{i=0}^{15} b_i \cdot 2^{-(16-i)} \tag{2}$$

Then a precision reduction of the fractional part is introduced to provide an approximated version for all values. The precision reduction ranges from 15 to 1 single bit. Since the integer part is not altered, each time the fractional part is reduced, the produced error ϵ follows Equation 3, where k is the number of bits removed from the fractional part and b_i is the value of the original bit removed.

$$\epsilon = \sum_{i=0}^{k} b_i \cdot 2^{-(16-i)} \tag{3}$$

For a value undergoing a precision reduction of k bits, the minimum and maximum errors are defined in Equation 4. Depending on k, the maximum error is when all removed b_i were equal to 1. While, the minimum error is 0 when all removed b_i were equal to 0.

$$\underbrace{0}_{\forall b_i=0} \leq \epsilon \leq \underbrace{\sum_{i=0}^{k} 2^{-(16-i)}}_{\forall b_i=1} = -\frac{1-2^k}{32768} \tag{4}$$

Looking at Equation 2, the v value can be expressed as $v-\epsilon$ as ϵ represents a value subtracted from the fractional part. The goal of applying such reduction is to deliver smaller numbers without compromising the computation accuracy too much.

C. The Interval Arithmetic Error Propagation Model

Using the error computed in Section II-B, it was possible to devise an Interval Arithmetic (IA) error propagation model. The foundation of IA is the ability to define mathematical operations over intervals. Intervals are supported by all basic mathematical operations, such as addition and multiplication [14]. Using the brackets notation, an interval is defined as in Equation 5, where v_{min} and v_{max} represent the two boundaries of the interval. The interval includes all values between the boundaries (them included) in the defined field, i.e., fixed-point values.

$$[v] \equiv [v_{min}, v_{max}] \tag{5}$$

Since Equation 4 expresses the errors introduced by the precision reduction as a range of errors, each number can be described as $v - [\epsilon]$. The v value can also become an interval when a range of values undergoing approximation is modeled. Thus, we defined a single interval arithmetic number ($|ian|$), expressing the potential values and the errors associated, as a pair of intervals $\{[v], [\epsilon]\}$. When $[v]$ and $[\epsilon]$ represent a single value or a single k reduction, the two boundaries of the interval are the same.

Moreover, based on Equation 4, it can be claimed that $|ian| \simeq [v] - [e]$, showing that the approximated value of an interval arithmetic number can be obtained from subtracting the range of errors that the precision reduction may produce ($[\epsilon]$) modeled as an always positive range in Equation 4, from the range of the original values ($[v]$).

It is crucial to notice that:

1) the errors come from Equation 4, thus they are strictly monotonic, i.e., $[\epsilon] \equiv [\epsilon_{min}, \epsilon_{max}]$. The same applies to $[v]$.
2) due to the nature of the precision reduction operation, the relationship between $[v]$ and $[\epsilon]$ is subtractive.

These considerations are critical to defining the mathematical operations required to translate all approximated values,

i.e., weights, thresholds, and reset values, into $\{[v],[\epsilon]\}$ intervals and being able to model the same operations as in the original model. The notation allows keeping the error separated from the value itself. This is a considerable difference from previous works such as [16] and supports the exploration capabilities of the modeling even further.

The following subsection applies the IA theory to model all mathematical operations required by subsection II-A.

1) Addition / Subtraction: The addition and the subtraction among two values, i.e., $|ian_1|$ and $|ian_2|$ can be defined using the linearity of the two mathematical operations and the monotonic shape of the ranges as in Equation 6 and Equation 7, resorting to basic interval operations among the two intervals in $|ian|$.

$$|ian_1| + |ian_2| = \{\underbrace{[v_{1_{min}} + v_{2_{min}}, v_{1_{max}} + v_{2_{max}}]}_{[v]}, \underbrace{[\epsilon_{1_{min}} + \epsilon_{2_{min}}, \epsilon_{1_{max}} + \epsilon_{2_{max}}]}_{[\epsilon]}\} \quad (6)$$

$$|ian_1| - |ian_2| = \{\underbrace{[v_{1_{min}} - v_{2_{max}}, v_{1_{max}} - v_{2_{min}}]}_{[v]}, \underbrace{[\epsilon_{1_{min}} - \epsilon_{2_{max}}, \epsilon_{1_{max}} - \epsilon_{2_{min}}]}_{[\epsilon]}\} \quad (7)$$

2) Multiplication: The multiplication is a particular case in which the common IA operation does not suit our modeling and is based on $|ian| \simeq [v] - [e]$ as in Equation 8.

$$\begin{aligned} |ian_1| \cdot |ian_2| &\simeq ([v_1] - [e_1]) \cdot ([v_2] - [e_2]) = \\ &\underbrace{[v_1] \cdot [v_2]}_{A} - (\underbrace{[v_1] \cdot [e_2]}_{B} + \underbrace{[v_2] \cdot [e_1]}_{C}) + \underbrace{[e_1] \cdot [e_2]}_{D} = \\ &\underbrace{[min(A), max(A)]}_{[v]}, \\ &\underbrace{[min\left(D - (B + C)\right), max\left(D - (B + C)\right)]}_{[\epsilon]} \end{aligned} \quad (8)$$

The four components A, B, C and D are fully developed in Equation 9 and demonstrate how the multiplication will see the influence of $[v]$ into $[\epsilon]$.

$$\begin{aligned} A &\rightarrow \{v_{1_{min}} v_{2_{min}}, v_{1_{min}} v_{2_{max}}, v_{1_{max}} v_{2_{min}}, v_{1_{max}} v_{2_{max}}\} \\ B &\rightarrow \{v_{1_{min}} \epsilon_{2_{min}}, v_{1_{max}} \epsilon_{2_{min}}, v_{1_{min}} \epsilon_{2_{max}}, v_{1_{max}} \epsilon_{2_{max}}\} \\ C &\rightarrow \{v_{2_{min}} \epsilon_{1_{min}}, v_{2_{max}} \epsilon_{1_{min}}, v_{2_{min}} \epsilon_{1_{max}}, v_{2_{max}} \epsilon_{1_{max}}\} \\ D &\rightarrow \{\epsilon_{1_{min}} \epsilon_{2_{min}}, \epsilon_{1_{min}} \epsilon_{2_{max}}, \epsilon_{1_{max}} \epsilon_{2_{min}}, \epsilon_{1_{max}} \epsilon_{2_{max}}\} \end{aligned} \quad (9)$$

3) Right Shift: The right shift is especially useful in the operation involving the exp_{decay}. Equation 10 summarizes its modeling for a $|ian|$.

$$|ian_1| \gg n = \{\underbrace{[v_{1_{min}} \gg n, v_{2_{max}} \gg n]}_{[v]}, \underbrace{[\epsilon_{1_{min}} \gg n, \epsilon_{2_{max}} \gg n]}_{[\epsilon]}\} \quad (10)$$

4) Comparison: One of the non-re-usable parts of the IA theory is the definition of the comparison between intervals. The model implements the decision-making of the $|ian_1| > |ian_2|$ operator, by rewriting each comparison as $|ian_1| - |ian_2| > 0$, thus resorting to the subtraction between two $|ian|$. Equation 11 reports the evaluation of such comparison in the IA modeling. Note that the same approach can be applied to all comparison operators.

$$\begin{aligned} &|ian_1| - |ian_2| > 0 \\ &\Rightarrow |ian_r| > 0 \\ &(v_{r_{max}} - \epsilon_{r_{max}}) - (v_{r_{min}} - \epsilon_{r_{min}}) > 0 \end{aligned} \quad (11)$$

Thanks to the monotonic property, once the difference is evaluated, we calculate a single interval using the $|ian|$ equivalence. If the difference between the minimum and the maximum (including the errors) is positive, the comparison is true (or elsewhere if different comparison operators are modeled). Thus the comparison results in a majority voting that allows a fast model evaluation, but it might introduce some errors. In fact if $(v_{r_{max}} - \epsilon_{r_{max}})$ and $(v_{r_{min}} - \epsilon_{r_{min}})$ have opposite signs, they model a scenario where the real comparison might be either true or false depending on the value in the interval.

III. Experimental Results

The trained SNN from [17] is used in this paper. The original network was trained using the MNIST dataset [21]. The MNIST dataset comprises 60,000 training images and 10,000 testing images of handwritten digits. The 784 pixels of each image (28x28 pixels) are converted into a sequence of spikes using a random Poisson process [22] with an average frequency corresponding to the numeric value in the pixel, resulting in 3500 spikes per image.

The complete SNN structure includes one layer with 400 neurons. The membrane potential of each neuron is reset to its rest potential at the end of each image. The network's outputs include: (i) the associated spike counter and (ii) the classification decision made by the final decision layer. Since we aim to assess the quality of the IA error-based modeling in this work, the final classification is not the main focus.

Figure 3 depicts the experimental workflow, for each k value of precision reduction. All data from the original SNN network are extracted and stored for further use. The red part of the workflow represents the original approximated network. All data undergo precision reduction for each k value, and the approximated SNN can run. The counting of the output spikes is then stored to be compared with the output of the IA model. The green blocks in Figure 3 describe the IA error-based modeling. Based on the network data (that feed the $[v]$ part of a $|ian|$), for each k, an automated procedure derives the $[\epsilon]$ part of each value to produce the $|ian|$ versions of each network data adequately. The process relies on 3 to estimate the $[\epsilon]$ for every network data undergoing the approximation. This process is faster than the approximation one in the red part of the figure because it is purely mathematical. Then the model

runs following the computational flow depicted in Figure 2 to compute the spike counter for each neuron. To allow a fair comparison of the final results, the sequence of input spikes used for the IA model should be in the same order as the sequence of input spikes used for the original SNN.

The experimental setup has been applied to 520 images of the test set, and results are reported in Table I. For each precision reduction (k), the table displays the range of errors in the counters and the percentage of wrong counters. Eventually, the last column considers the classification done using the counters computed from the model compared with the one performed by the original network.

Reduction (k-bits)	Errors			Wrong Counters (%)	SNN Accuracy Comparison
	Min	Max	AVG		
1	-1,00	1,00	0,0009	0,28	Same
2	-1,00	1,00	0,0011	0,33	Same
3	-2,00	1,00	0,0003	0,58	Same
4	-2,00	1,00	-0,0010	0,86	Same
5	-3,00	2,00	0,0000	1,60	Same
6	-3,00	1,00	-0,0010	2,38	Same
7	-2,00	2,00	-0,0030	3,44	Same
8	-2,00	2,00	-0,0031	4,77	Same
9	-2,00	2,00	-0,0070	5,31	Same
10	-2,00	2,00	-0,0074	6,16	Same
11	-2,00	2,00	-0,0107	6,86	Same
12	-2,00	2,00	-0,0299	7,34	Same
13	-3,00	3,00	-0,0415	7,12	Same
14	-4,00	2,00	-0,0431	5,38	Same
15	-3,00	2,00	-0,0256	2,49	Same

TABLE I: IA Model single k results on spike counters and related network accuracy in prediction

It is important to note that the error columns in Table I show the minimum, maximum, and average error between the output spike count obtained by the IA-based model and the output spike count obtained by executing the original approximated network. Therefore, a negative value indicates that the model is producing (hence counting) fewer spikes in the IA model, while a positive value indicates otherwise.

As displayed by the Errors columns, the model does not show many discrepancies from the original SNN. In most cases, the minimum and maximum errors stay closer to 2 or 3. The worst case happens when the precision is down to 14 bits removed where a -4 is shown. From a stability perspective, the `AVG` column suggests that the error amount is generally less than the minimum and maximum, most of them generating a few spikes less in the IA model (the negative sign on most averages). It is also interesting that a few counters are wrong at the end of the IA model elaboration. The percentage increases while increasing the k reduction, confirming the hypothesis made in subsection II-C that the model simplifies the comparison by preventing some spikes from being fired. The apparent improvement at $k = 14$ is only due to the fact that the original network accuracy is already lost at $k = 10$, as reported in [17].

Eventually, the network accuracy comparison always shows the same results because of the structure of the network decision model: the 400 counters are split into groups of 40 counters, and the classification is made by selecting the group showing the highest number of counts. In this context, the percentage of wrong counters is not altering the classification significantly.

Once the model's correctness for single errors is verified, we investigate the opportunity of predicting the impact of approximation for a range of k_{min} to k_{max} precision reductions. This reduces the time of exploration by considering more reductions per time. The results for a statistically meaningful set of 30 images are shown on Table II. The definitions of columns follow Table I except the wrong counters percentage. The reason is that this exploration has been compared with the original network outputs when applied the high reduction k. Since ranges of errors are considered, we do not seek exact correspondence of counters number but the capability of predicting the correct behavior. Though the average error in calculating the counters is higher than before, still the time required to run the evaluation is $\frac{1}{5}th$ before, since each run considers 5 k values per time. However, the network classification accuracy is untouched.

Reduction (k_{min} to k_{max}-bits)	Errors			SNN Accuracy Comparison
	Min	Max	AVG	
1 to 5	0	10	3,1777	Same
6 to 10	0	10	3,1950	Same
11 to 15	0	10	3,2776	Same

TABLE II: IA Model results on ranges of k_{min} to k_{max} reductions. The time required to run the evaluation is $\frac{1}{5}th$ before, since each run considers 5 k values per time.

Concluding from Table I and Table II and the fact that the original network accuracy is already lost at $k = 10$, as reported in [17], the most aggressive approximation while not sacrificing the SNN accuracy can be considered as $k = 10$.

IV. Conclusion

We introduced a model for exploring different approximated versions of a spiking neural network and predicting the impact of approximation on the accuracy of the computation results. For this purpose, we modeled the computation flow of the SNN using Interval Arithmetic to quickly evaluate the impact of approximation in terms of loss in computation accuracy without executing the network each time.

We conducted the experiments for a trained SNN executed in inference mode to find whether the predicted network classification accuracy for each approximated version remains the same as the network classification accuracy obtained by running the original approximated version. Then, we also verified that some less refined exploration could be carried out, saving time without losing sight of the quality of the application. Experimental results comparing our model to the original networks confirm the quality of the approach.

For future work, we plan to extend the model to other neural networks using different data sets other than MNIST and employ different AxC techniques to investigate this approach's opportunities further. However, it is important to note that our goal was to quickly analyze the impact of approximation

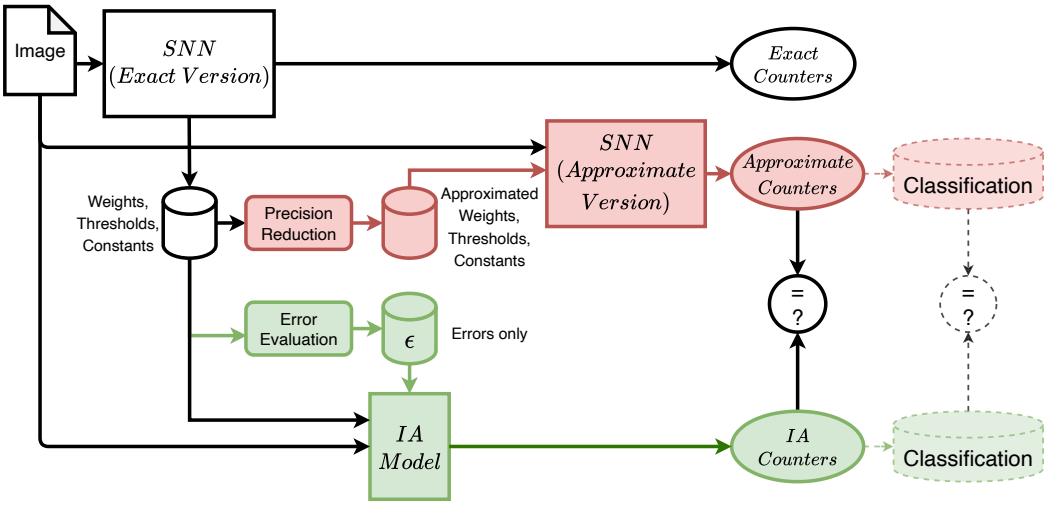

Fig. 3: Experimental workflow. For each image and k precision reduction, the red part evaluates the real values running the full network, while the green part depicts the model where only errors are computed for each k.

on SNN computations accuracy, not proving the benefits of applying AxC techniques on the gains in performance. It would be nice to utilize our approach to compare the impact of different AxC techniques on the computation accuracy as well as their impact on gains in performance, and finally weigh these against each other to choose the best AxC techniques.

Acknowledgment

This work has received funding from the APROPOS project in the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 956090.

References

[1] P. Stanley-Marbell, A. Alaghi, M. Carbin, E. Darulova, L. Dolecek, A. Gerstlauer, G. Gillani, D. Jevdjic, T. Moreau, M. Cacciotti, A. Daglis, N. E. Jerger, B. Falsafi, S. Misailovic, A. Sampson, and D. Zufferey, "Exploiting errors for efficiency: A survey from circuits to applications," *ACM Comput. Surv.*, vol. 53, no. 3, jun 2020. [Online]. Available: https://doi.org/10.1145/3394898

[2] D. Wu and J. S. Miguel, "Special session: When dataflows converge: Reconfigurable and approximate computing for emerging neural networks," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 9–12.

[3] T. Ayhan and M. Altun, "Approximate fully connected neural network generation," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018, pp. 93–96.

[4] Z. Peng, X. Chen, C. Xu, N. Jing, X. Liang, C. Lu, and L. Jiang, "Axnet: Approximate computing using an end-to-end trainable neural network," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[5] Z. Wang, M. A. Trefzer, S. J. Bale, and A. M. Tyrrell, "Approximate multiply-accumulate array for convolutional neural networks on fpga," in *2019 14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2019, pp. 35–42.

[6] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, "Neural networks designing neural networks: Multi-objective hyper-parameter optimization," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[7] E. Dupuis, D. Novo, I. O'Connor, and A. Bosio, "On the automatic exploration of weight sharing for deep neural network compression," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1319–1322.

[8] M. Traiola, A. Savino, and S. Di Carlo, "Probabilistic estimation of the application-level impact of precision scaling in approximate computing applications," *Microelectronics Reliability*, vol. 102, p. 113309, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271418309442

[9] A. Savino, M. Traiola, S. D. Carlo, and A. Bosio, "Efficient neural network approximation via bayesian reasoning," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2021, pp. 45–50.

[10] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225 134–225 180, 2020.

[11] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[12] R. V. W. Putra and M. Shafique, "Fspinn: An optimization framework for memory-efficient and energy-efficient spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3601–3613, 2020.

[13] ——, "Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments," *arXiv preprint arXiv:2103.00424*, 2021.

[14] "Ieee standard for interval arithmetic," *IEEE Std 1788-2015*, pp. 1–97, 2015.

[15] H. Okada, T. Matsuse, T. Wada, and A. Yamashita, "Interval ga for evolving neural networks with interval weights and biases," in *2012 Proceedings of SICE Annual Conference (SICE)*, 2012, pp. 1542–1545.

[16] J. P. Turner and T. Nowotny, "Arpra: An arbitrary precision range analysis library," *Frontiers in Neuroinformatics*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fninf.2021.632729

[17] A. Carpegna, A. Savino, and S. Di Carlo, "Spiker: Fpga-optimized hardware accelerator for spiking neural networks," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022, pp. 1–6.

[18] D. Gerlinghoff, Z. Wang, X. Gu, R. S. M. Goh, and T. Luo, "E3ne: An end-to-end framework for accelerating spiking neural networks with emerging neural encoding on fpgas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3207–3219, 2022.

[19] T. Spyrou, S. A. El-Sayed, E. Afacan, L. A. Camunas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, "Reliability analysis of a spiking neural network hardware accelerator," in *2022 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2022, pp. 370–375.

[20] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, p. e47314, Aug. 2019.

[21] C. C. Yann LeCunn and C. J. Burges. The mnist database. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[22] D. Heeger, "Poisson model of spike generation," 10 2000.