Received: 23 December 2020 | Revised: 15 July 2021 | Accepted: 1 August 2021

DOI: 10.1002/cta.3128

ORIGINAL PAPER

WILEY

Simplicial computation: A methodology to compute vector–vector multiplications with reduced complexity

P. Julian[1,2] | A. G. Andreou[3] | M. Villemur[1] | N. Rodriguez[1]

[1]Embedded Systems Division, Silicon Austria Labs, Linz, Austria

[2]IIIE-DIEC, UNS-CONICET, Bahia Blanca, Argentina

[3]ECE Department, The Johns Hopkins University, Baltimore, Maryland, USA

Correspondence
P. Julian, Embedded Systems Division, Silicon Austria Labs, Linz, Austria.
Email: pedro.julian@silicon-austria.com

Funding information
Fondo para la Investigación Científica y Tecnológica; PICT 2016, Grant/Award Number: 2009; PICT 2017, Grant/Award Number: 2526; Universidad Nacional del Sur; PGI 2019, Grant/Award Number: 26/K086; ANPCyT of Argentina; UNS

Summary

In this paper, we propose the use of the simplicial algorithm, originally proposed to implement piecewise-linear functions, to compute a digital vector–vector multiplication (VVM) without multiplications. The main contributions of the proposed methodology are (a) an improved error propagation with respect to parameter quantization; (b) a more efficient digital implementation with respect to area, energy, and speed in the case of large number of inputs; and (c) an intrinsic capability to produce multivariable nonlinear processing. We show that when quantization of inputs and parameters are considered, the simplicial method achieves the same accuracy with fewer representation bits for the parameters, assuming the same quantization for the inputs. Actually, in the particular case of a large number of inputs, the simplicial method needs half the number of parameter bits of a linear combination plus one.

KEYWORDS

linear combination, piecewise-linear functions, vector–vector multiplication

1 | INTRODUCTION

Vector–vector or matrix–vector multiplication (VVM/MVM) are pervasive operations and the basis of the most relevant and commonly used algorithms in signal processing:[1,2] fast Fourier transforms (FFTs), convolutions, digital filters, and neural networks are some of the most prominent examples. In particular, deep and convolutional neural networks (abbreviated as DNN and ConvNets, respectively), are nowadays receiving considerable attention in the area of machine learning due to its high efficacy in classification tasks. The major workload in these large networks is due to two types of vector–vector operations: sliding convolutions and static weighted combinations. For example, the well-known network AlexNet[3] has five convolutional layers and three fully connected layers; the convolutional layers have 2.97M single-precision floating-point (SPFP) parameters, that is, 32b weights, and require 775M operations to compute, while the three linear layers have 57M weights and require 57M operations. Dedicated processors have been reported in the literature[4-6] which are capable of executing between 1 and 50 convolutional layers (most time consuming) per second,[5] depending on the network, at tens of milliwatts of power consumption. Executing these networks in mobile devices is still a challenge and acts as a strong driver for the exploration of alternatives to reduce computation time, energy, and storage.[7] One possibility is to reduce the word length of the parameters. Indeed, different quantization methods have been proposed along this line at the expense of some degradation in the resulting accuracy (see other studies[8-10] and the references therein). For example, Wu et al[11] subdivides the weights into subgroups and selects a reduced number of representative weights to store. The inputs (also called activations) are multiplied with the corresponding representative weights and then the additions are performed, which leads to a 4× speedup factor and a 10× reduction in storage with an increase of 2.5% in classification error. This technique, called weight sharing, together

with sparse matrix vector multiplication, has been successfully implemented on chip in Han et al,[12] where a weight data reduction of 96% and a computation energy reduction by a factor between 3 and 7 has been shown. Based on this concept, Garland and Gregg[13] proposed a counting and accumulating scheme, which avoids products, resulting in 66% fewer gates and 70% less total power than a standard multiply–accumulate (MAC) based implementation. The work in Zhou et al[8] shows similar performance in network accuracy when (only) weights are quantized as powers of two with 5, 4, and even 3 bits, with respect to the original network with 32 bit-width weights. A second possibility is to also quantize activations: Rastegari et al.'s[14] study was one of the first works to quantize weights and activations simultaneously, while Jung et al[9] proposed a quantization with an adjustable interval that achieves full precision with 4 bits of weights and activations. Wu et al[15] proposed a variable precision scheme with different weight ({1, 2, 3, 4, 8, 32}) and activation ({3, 4, 8, 32}) bits for different layers of a given network, and showed similar performance to full precision models with a compression between 30% and 40%. Approximate arithmetic approaches without multiplications, based on barrel shifters and low power Wallace trees were proposed in Park et al.[10]

We are interested in large-scale VLSI-friendly alternative VVM digital implementations where the word length of inputs and parameters must be minimized to satisfy the increasing needs of accuracy, storage, and energy demanded by machine learning and artificial intelligence applications in portable devices. In this paper, we propose the simplicial[16] piecewise linear (PWL) computation method described in Julian et al[17] as an efficient alternative for neuromorphic digital computing, especially for large-scale systems. The main contributions of the proposed methodology are (a) an improved error propagation with respect to parameter quantization; (b) a more efficient digital implementation with respect to area, energy, and speed in the case of large number of inputs; and (c) an intrinsic capability to produce multivariable nonlinear processing. At this point, it is convenient to briefly elaborate on each one of these points.

In order to put these aspects in evidence, the approach followed throughout the paper consists of taking a standard VVM and design an equivalent simplicial PWL interpolation for comparison purposes. When no quantization errors are considered, the numerical results of both expressions are naturally the same. However, in the presence of quantization errors, we demonstrate that the simplicial interpolation is able to achieve the same accuracy as the VVM but using fewer representation bits for the parameters (assuming, of course, an equal number of representation bits for the inputs). Actually, when the number of inputs N is large, the equivalent number of parameter bits of the simplicial algorithm is half the number of bits of the VVM parameters, plus one. This is a novel and fundamental result, which not only adds to the representation properties of piecewise-linear functions[16,18,19] but, needless to say, also has a direct impact on the associated digital implementation, since fewer parameter bits translate into less area, energy, and propagation delay.

Regarding the second point, the simplicial interpolation can be computed without the need of multiplications, only by using a fixed number of additions that depends on the number of bits of the inputs, but not on the number of inputs. When the number of inputs is large, the simplicial computation requires less energy, area, and time than a classical VVM. In order to show this, we compare the corresponding digital architectures and evaluate both at a fundamental level, in terms of the number of standard CMOS gates and operation cycles. In addition, we propose a modified comparator for the simplicial algorithm that minimizes activity and produces a significant reduction in energy with respect to previous realizations.[20,21]

As explained in Julian et al,[17] the simplicial PWL algorithm can be decomposed into two operations: a first one, where the inputs are sorted and successive differences are taken (input encoding), and a second one where a linear interpolation is performed with the corresponding parameters. Therefore, the simplicial PWL algorithm can naturally implement the family of nonlinear filters known as order statistic (OS) filters.[22] These filters implement VVM on rank-ordered inputs, that is, inputs belonging to domains like[1]

$$S = \left\{ \mathbf{x} \in \mathbb{R}^N : 0 \le x_1 \le x_2 \le \ldots \le x_N \le 1 \right\}. \tag{1}$$

The most famous OS filter is the median filter, which is a selection-type filter, since the output coincides with only one of the inputs.[23] Another example is the alpha-trimmed mean filter and the max(min) filters. OS filters can also operate on weighted inputs and output a weighted sum, leading to the so called general OS filters.[24] These operations can be performed with the same architecture that produces the simplicial interpolation therefore enabling a wide variety of nonlinear operators. One example widely used in image processing is the family of morphological operations.[25] Indeed, we have recently reported[21] a VLSI implementation of a 48 × 48 simplicial cellular neural network array based

[1]Indeed, (1) is the definition of a simplex and gives the name to the simplicial representation

on this approach that achieves a remarkable efficiency of 3.4 fJ per operation in a 55-nm CMOS technology. For illustration purposes, we show an example where the proposed architecture implements a state-of-the-art morphological DNN ([26]) for the recognition of city maps.

The paper is organized as follows. Section 2 describes the computational method and the operations that must be carried out to compute a simplicial interpolation versus a VVM. Section 3calculates the propagation of errors to the output due to input and parameter quantization. Section 4 analyzes the energy required by a digital architecture. Finally, Section 6 contains the conclusions.

2 | COMPUTATIONAL METHODOLOGY AND PROPERTIES

2.1 | Simplicial computation

A simplicial PWL function is illustrated in Figure 1, where one simplex (a triangle in $\mathbb{R}^2$ and its generalization in $\mathbb{R}^N$) is defined by vertices $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$. The function in this simplex is uniquely defined by the parameters c_1, c_2, and c_3. In other simplices, the function is defined by other parameters, thereby allowing the construction of linear, nonlinear, and even discontinuous PWL functions. Our approach utilizes two hierarchy levels. A memory stores the parameters, which are the values of the VVM at the vertices of a simplex where $\mathbf{x}$ belongs to. The memory passes these values as parameters to a computation unit that performs an interpolation among the parameters and produces the result, namely, the simplicial interpolation.

Let us consider a weighted combination of terms in the following form:

$$y = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{N} w_i x_i, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^N$ is a vector belonging to the hypercube $[0,1]^N$. As shown in Chien and Kuh,[16] any interior point of the hypercube can be written as a convex combination of at most $N+1$ vertices[2] $\mathbf{v}_j$, as follows:

$$\mathbf{x} = \sum_{j=1}^{N+1} \mu_j \mathbf{v}_j. \tag{3}$$

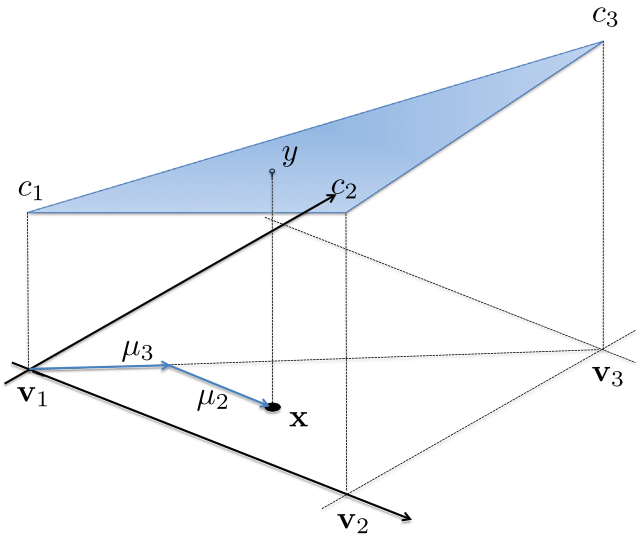

FIGURE 1 A linear function over a simplicial division of a domain with two simplices

[2]A vertex $\mathbf{v}_j$ is defined as the sum of j unit vectors, $\mathbf{e}_i$, $i \in \{1, \ldots, N\}$.

The convex combination implies that the values μ_j satisfy $\mu_j \geq 0$ and

$$\sum_{j=1}^{N+1} \mu_j = 1. \tag{4}$$

The ordered sequence of $N+1$ vertices $\{\mathbf{v}_1, \ldots, \mathbf{v}_{N+1}\}$ defines a particular simplex contained in the hypercube. In general, a hypercube in $\mathbb{R}^N$ possesses $N!$ simplices. As an example, Figure 1 illustrates the simplex in $\mathbb{R}^2$ defined by the vertices $\mathbf{v}_1 = [1,1]$, $\mathbf{v}_2 = [1,0]$, and $\mathbf{v}_3 = [0,0]$. The interior point $\mathbf{x} = [0.5, 0.2]$ can thus be written as $\mathbf{x} = \mu_1\mathbf{v}_1 + \mu_2\mathbf{v}_2 + \mu_3\mathbf{v}_3$, where $\mu_1 = 0.2$, $\mu_2 = 0.3$, and $\mu_3 = 0.5$.

The μ values in the simplicial expression play the role of the coordinate values x. If we replace (3) into (2), we obtain the expression of function (2) as a function of the μ values:

$$y = \mathbf{w}^T \sum_{j=1}^{N+1} \mu_j \mathbf{v}_j = \sum_{j=1}^{N+1} \mu_j (\mathbf{w}^T \mathbf{v}_j) = \sum_{j=1}^{N+1} \mu_j c_j = \mu^T \mathbf{c}, \tag{5}$$

where $c_j = \mathbf{w}^T \mathbf{v}_j$, $j = 1, \ldots, N+1$ can be interpreted as new parameters, corresponding to the simplicial expression. In fact, c_j are the values of (2) at the simplex vertices. For instance, if $\mathbf{w}^T = [1,1]$ corresponding to function $y = x_1 + x_2$, then $c_1 = [1,1] \times \mathbf{v}_1 = 2$, $c_2 = [1,1] \times \mathbf{v}_2 = 1$, $c_3 = [1,1] \times \mathbf{v}_3 = 0$. Evaluation of y at $\mathbf{x} = [0.5, 0.2]$ results in $y = \mu_1 c_1 + \mu_2 c_2 + \mu_3 c_3 = 0.2 \times 2 + 0.3 \times 1 + 0.5 \times 0 = 0.7$.

The values μ_i can be calculated by first sorting the input components $\{x_1, \ldots, x_N\}$ in ascending order, namely, $\{\hat{x}_1, \ldots, \hat{x}_N\}$ and then taking differences as indicated in the Algorithm 1.

Algorithm 1 Simplicial algorithm

```
x̂_1, ..., x̂_n ← sort{x_1, ..., x_n}
μ_1 ← x̂_1
y ← μ_1 c_1
if i ≤ N then
    μ_i ← x̂_i − x̂_{i−1}
    y ← y + μ_i c_i
end if
μ_{N+1} ← 1 − x̂_N
y ← y + μ_{N+1} c_{N+1}
```

In matrix form, the vector with the firs N components of $\mu = [\mu_1 \ldots \mu_N]^T$ can be written as $\mu = D \times \mathbf{x}$, where D is the matrix defined as

$$D = \begin{bmatrix} 1 & 0 & \ldots & & 0 \\ -1 & 1 & 0 & \ldots & 0 \\ & & \vdots & & \\ 0 & \ldots & 0 & -1 & 1 \end{bmatrix}. \tag{6}$$

2.2 | Numerical complexity

Let us assume now that x_i and μ_i are quantized using q bits into $Q = 2^q$ different levels in order to evaluate the total number of operations required. For this purpose, we can assume that $x_i = k_i \Delta l$, $\mu_i = \kappa_i \Delta l$, where $\Delta l = 1/Q$ and $k_i, \kappa_i \in [0, Q-1]$ are integers. At first sight, expression (5) requires $N+1$ sums and N products. A closer inspections reveals that the simplicial expression can be written as follows:

$$y = \sum_{j=1}^{N+1} \mu_j c_j = \sum_{j=1}^{N+1} \Delta l (\kappa_j c_j) = \Delta l \sum_{j=1}^{N+1} \left(\sum_{k=1}^{\kappa_j} c_j \right), \tag{7}$$

where we have used the fact that $\kappa_j c_j = \sum_{k=1}^{\kappa_j} c_j$. Every term $\sum_{k=1}^{\kappa_j} c_j$ requires $\kappa_j - 1$ additions; therefore, (7) requires a total number of sums given by

$$S_{\text{SIMP}} = N + \sum_{k=1}^{N+1} (\kappa_k - 1) = -1 + \sum_{k=1}^{N+1} \kappa_k. \tag{8}$$

Due to (4), $\sum_{k=1}^{N+1} \mu_k = \Delta l \sum_{k=1}^{N+1} \kappa_k = 1$ so that

$$\sum_{k=1}^{N+1} \kappa_k = \frac{1}{\Delta l} = Q. \tag{9}$$

Therefore, the simplicial expression can be implemented using only

$$S_{\text{SIMP}} = Q - 1 \tag{10}$$

additions, assuming that the inputs have already been sorted to produce the μ values (Figure 2).

Note that if the number of inputs N is greater than Q, then at least $N - Q$ inputs will have the same value, and there will be at most Q values μ_i different from zero. In that case, the number of *nonzero* terms in the summation (7) is at most Q and becomes independent of N. This is illustrated in the example of Figure 3, where we assume there are only

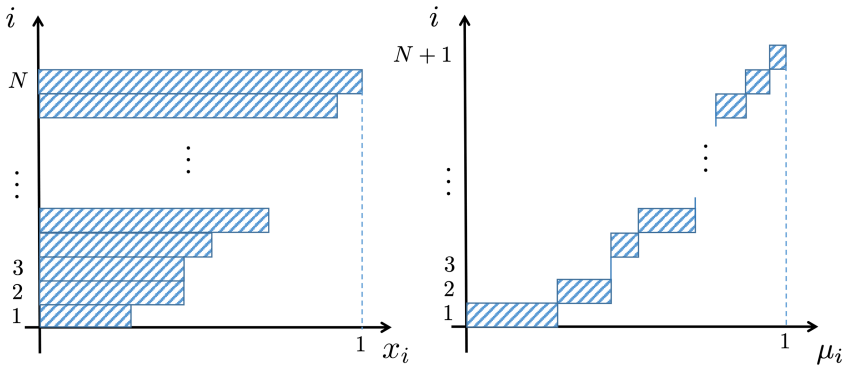

FIGURE 2 Graphical representation of (sorted) input values, x_i (left) versus the μ_i components (right). From here, it can be clearly appreciated that the number of elements to be summed in the simplicial expression is $Q - 1$, as given by (10)

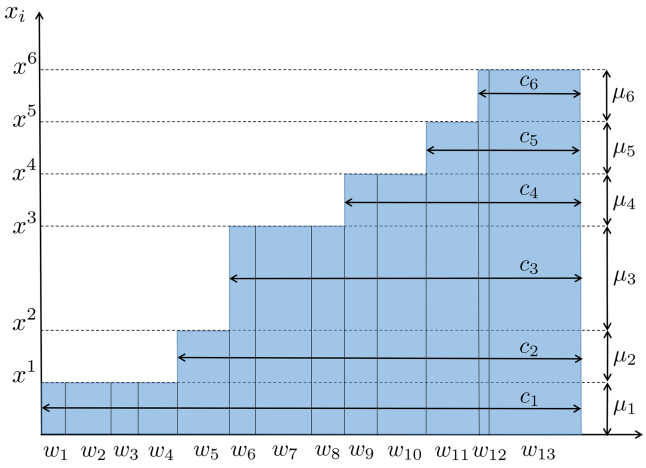

FIGURE 3 Illustration of the simplicial interpolation versus a VVM. If the number of *different* numerical values x^i does not change, the number of horizontal rectangles corresponding to pairs (c_i, μ_i) remains constant; even if the number of terms, given by the number of pairs (x_i, w_i), increases

$Q = 6$ different numerical values x^j for each component of x (vertical axis) and $N = 13$ parameters w_i (horizontal axis). The value of the linear combination y is the shaded area, which can be alternatively interpreted as the union of the many columns of width w_i and height given by the corresponding value of x, or the union of the rows of width c_i and height given by the corresponding value of μ. Note that only six values μ_i are different from zero, and the summation only requires six terms regardless of the number of parameters w_i.

If we assume that p is the number of bits used to quantize parameters c_i, the dynamic range—in bits—of output y is

$$DR(y) = q + p, \tag{11}$$

which is also independent of N.

On the other hand, the product in the VVM (2) requires $N - 1$ additions and N products, and assuming that r is the number of bits used to quantize parameters w_i, its associated dynamic range (2) is

$$DR(y) = \log_2(N) + q + r. \tag{12}$$

For equal number of quantization levels in their respective parameters and inputs, the two representations are different: the VVM has an additional $\log_2(N)$ number of bits. In the particular case where $p = \log_2(N) + r$, both expressions are identical and have exactly the same dynamic range.

3 | QUANTIZATION EFFECTS

In this section, we consider a weighted combination of terms as in (2) and an associated simplicial representation (5), where the parameters and inputs are full precision numbers. This could be the situation after an optimization algorithm has been applied to the given set of data to produce an optimal fit. We are interested in the analysis of the errors that occur in both representations when we quantize the inputs and parameters to produce a digital implementation.

3.1 | Parameter quantization

Let us assume first that the input is represented with no error using q bits and we only want to analyze the effect of parameter quantization on the output error.

In the case of a VVM, the parameters w_i, $i = 1,...,N$, are represented as $w_i = \hat{w}_i + \delta w_i$, where $\hat{w}_i$ is quantized with r bits. Without loss of generality we assume that x_i and w_i have uniform distributions in the ranges $[-R_x/2, R_x/2]$ and $[-R_w/2, R_w/2]$. The output is given by

$$y = \sum_{i=1}^{N} \hat{w}_i x_i + \delta w_i x_i, \tag{13}$$

and the error is

$$\delta y = \sum_{i=1}^{N} \delta w_i x_i, \tag{14}$$

which has zero mean and a variance equal to

$$var(\delta y) = \sum_{i=1}^{N} var(\delta w_i x_i). \tag{15}$$

As w_i is represented with r bits in the interval $[-R_w/2, +R_w/2]$, δw_i lies in the interval $2^{-r} \times [-R_w/2, +R_w/2]$. Assuming that x_i and w_i are not correlated and have zero mean, the error variance can be written as follows:

$$var(\delta y) = \sum_{i=1}^{N} var(\delta w_i) var(x_i) = \frac{(R_w 2^{-r})^2}{12} \sum_{i=1}^{N} var(x_i). \tag{16}$$

Since x_i has uniform distribution, $\sum_{i=1}^{N} x_i$ has an Irwin–Hall distribution with zero mean and variance $R_x^2 N/12$. Therefore, (16) can be written as

$$var(\delta y) = \frac{(R_w 2^{-r})^2}{12} \frac{R_x^2 N}{12}, \tag{17}$$

and the standard deviation expression is

$$\sigma(\delta y) = \frac{R_w R_x \sqrt{N}}{12} \times 2^{-r}. \tag{18}$$

The range of y is the summation of the product of two random variables with uniform distribution, so it has a normal distribution with zero mean and standard deviation:

$$\sigma(R(y))_{Lin} = \frac{R_x R_w}{12} \sqrt{N}. \tag{19}$$

Then, the output range can be approximated with k-σ ($k \geq 3$) confidence as follows:

$$R(y) \approx 2k \times \frac{R_x R_w}{12} \sqrt{N}, \tag{20}$$

and the standard deviation of the error relative to the output range is given approximately by

$$\boxed{\frac{\sigma(\delta y)}{R(y)}_{Lin} \approx \frac{1}{2k} \times 2^{-r}}. \tag{21}$$

In the simplicial case, the parameters are represented as $c_j = \hat{c}_j + \delta c_j$, $j = 1, \ldots, N+1$, where $\hat{c}_j$ has p bits, and δc_j is the quantization error; the inputs are represented as $\mu_j = \hat{\mu}_j + \delta \mu_j$, where $\hat{\mu}_j$ has q bits and $\delta \mu_j$ is the quantization error. As we mentioned, for this analysis, the input is discrete but has no errors, so that $\delta \mu_j = 0$. The quantized values $\hat{\mu}_j$, $j = 1, \ldots, N+1$, are obtained by first quantizing each x_j, then sorting them and lastly taking the difference, that is, $\hat{\mu} = D \times \hat{\mathbf{x}}$, where D was defined in (6).[3]

The function can be Taylor-expanded in a neighborhood of $(\hat{\mu}, \hat{\mathbf{c}})$ as

$$y = \langle \hat{\mu}, \hat{\mathbf{c}} \rangle + \sum_{j=1}^{N+1} \left. \frac{\partial y}{\partial \mu_j} \right|_{\hat{\mu}, \hat{\mathbf{c}}} \delta \mu_j + \sum_{j=1}^{N+1} \left. \frac{\partial y}{\partial c_j} \right|_{\hat{\mu}, \hat{\mathbf{c}}} \delta c_j = \langle \hat{\mu}, \hat{\mathbf{c}} \rangle + \sum_{j=1}^{N+1} \hat{c}_j \delta \mu_j + \sum_{j=1}^{N+1} \hat{\mu}_j \delta c_j. \tag{22}$$

Considering that there are only $T \triangleq \min\{N, Q\}$ values $\hat{\mu}_j$ different from zero and that $\delta \mu_j = 0$, for every $j = 1, \ldots, N+1$, the representation error is

$$\delta y = \sum_{j=1}^{T} \hat{\mu}_j \delta c_j. \tag{23}$$

The values $\hat{\mu}_j$, $j = 1, \ldots, N+1$, are the differences between consecutive values of the already sorted (in increasing order) sequence of values $\hat{x}_j$, $j = 1, \ldots, N$; therefore, they have a triangular distribution. If the number of quantization levels is greater than the number of inputs, that is, $Q >> N$, the distribution of $\hat{\mu}_j$ has a mean $E(\hat{\mu}_j) = 1/N$ and

[3]Another possibility would be quantizing the distribution of μ_j.

$var(\hat{\mu}_j) = 1/N^2$. On the other hand, if the number of inputs is large, that is, $N >> Q$, there will be a large number of $\hat{\mu}_j$ values equal to zero and at most Q values with a mean of $E(\hat{\mu}_j) = 1/Q$ and a variance $var(\hat{\mu}_j) = 1/Q^2$. Accordingly, the error variance can be summarized as

$$var(\delta y) = \sum_{j=1}^{T} \frac{1}{T^2} var(\delta c_j) = \frac{1}{T^2}(2^{-p}R_c)^2 \frac{T}{12} = \frac{(2^{-p}R_c)^2}{12T}. \tag{24}$$

The coefficients c_i coincide[4] with the value of y evaluated at the vertices of the domain, so that R_c can be calculated as the range of the output:

$$R_c \approx 2k \frac{R_w R_x \sqrt{N}}{12}. \tag{25}$$

Considering this, the standard deviation is given by

$$\sigma(\delta y) \approx \frac{2kR_w R_x}{\sqrt{12}^3} \sqrt{\frac{N}{T}} \times 2^{-p}, \tag{26}$$

and the standard deviation relative to the output is

$$\boxed{\frac{\sigma(\delta y)}{R_y}_{Simp} \approx \frac{1}{\sqrt{12T}} \times 2^{-p}}. \tag{27}$$

Equations (24) and (27) provide the evidence behind the improved accuracy of the simplicial computation. The variance of the error in (24) is proportional to T due to the number of terms in the summation and inversely proportional to T^2 due to the variance of $\hat{\mu}_j$. The net effect is a variance inversely proportional to T.

Note that while the relative variance of the error in the parameters (considering $p = r$) is the same, that is,

$$\frac{var(\delta w_i)}{R_w^2} = \frac{var(\delta c_i)}{R_c^2} = \frac{2^{-p}}{12} = \frac{2^{-r}}{12}, \tag{28}$$

the relative variance of μ is $1/T^2$, much smaller than $var(x_i)/R_x^2 = 1/12$. The net result is a reduction of the error due to parameter quantization when T (the minimum between the number of inputs and the quantization levels) increases. This is a distinctive advantage of the simplicial input encoding, which represents the inputs by using relative differences of monotonically increasing values.

Figure 4 illustrates the parameter error propagation in both cases. Figures 5 and 6 reproduce Figure 4for $N = 32$, $q = 3$, $p = r = 3$. In Figure 5, the VVM is expressed geometrically as the summation of vertical rectangles of area equal to $w_i x_i$. On the other hand, Figure 6 expresses the simplicial combination as the summation of horizontal rectangles of area equal to $\mu_i c_i$. Notice that Figures 5B and 6B are identical because $\hat{x}_i$ is represented with no error by $\hat{\mu}_i$.

Example 3.1. A numerical simulation was performed using random values of inputs and parameters for different values of $N = 2, 4, 8, \ldots, 2048$. Inputs and parameters were uniformly distributed in the interval $[-1, 1]$, assuming no error in the inputs ($q = 6$) and sweeping the parameter accuracy (p, r) from 2 bits to 10 bits. For every combination of the number of inputs and parameter bits, 10,000 random values were evaluated. Figure 7 shows the relative standard deviation of the error for both the simplicial interpolation and the VVM.

[4]The parameters w_i have a uniform distribution, and the coefficients c_i are summations of subsets of them, so, in general, their distribution will be Gaussian. The distribution of the error δc_i can be considered uniform to compute a bound on the total error.

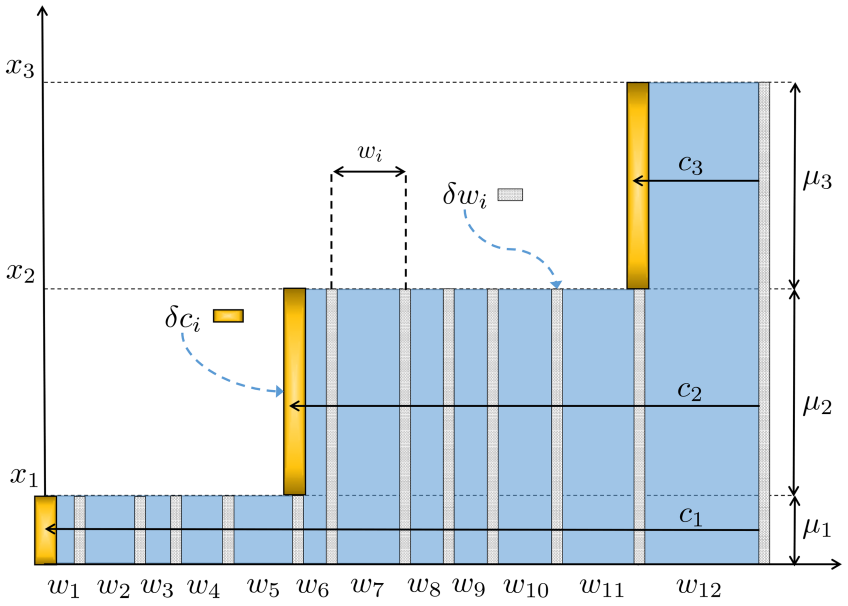

FIGURE 4 Parameter error propagation in the simplicial interpolation versus a VVM. Thin dot-filled white bars indicate errors due to quantization in w_i terms; thin shaded bars indicate errors due to quantization in the c_i terms

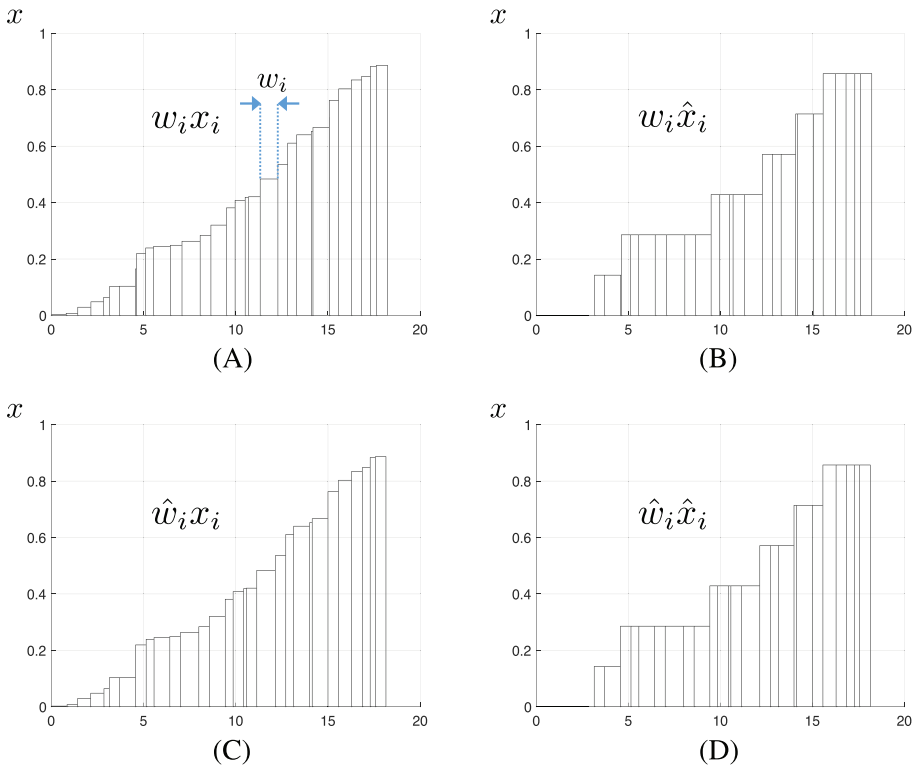

FIGURE 5 Illustration of components in a VVM: (A) no error; (B) quantization in x_i; (C) quantization in w_i; (D) quantization in both terms. $N = 32$, $q = 3$, $p = r = 3$

Further analysis shows that if the number of inputs is smaller than the number of quantization levels of the input ($N < Q$), then $T = N$ and the standard deviation (26) is independent of N:

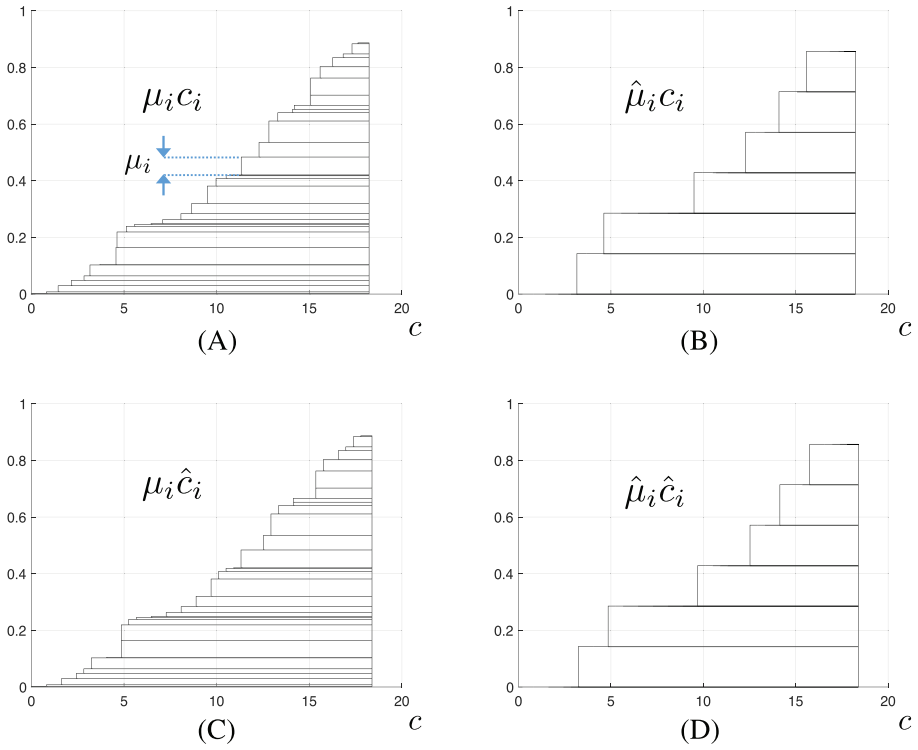

FIGURE 6 Illustration of components in the simplicial combination: (A) no error; (B) quantization in μ_i; (C) quantization in c_i; (D) quantization in both terms. $N = 32$, $q = 3$, $p = r = 3$

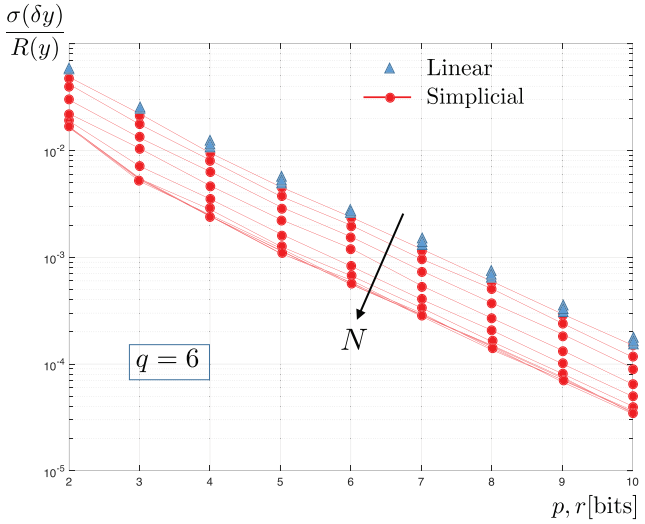

FIGURE 7 Numerical simulation of the relative standard deviation of the simplicial interpolation and the VVM for several values of $N = 2, 4, 8, \ldots, 2048$. Input quantization is $q = 6$ bits

$$\sigma(\delta y) \approx \frac{2kR_w R_x}{\sqrt{12^3}} \times 2^{-p}. \tag{29}$$

In this case, the relative standard deviation is given by

$$\frac{\sigma(\delta y)}{R_y} \approx \frac{1}{\sqrt{12N}} \times 2^{-p}, \tag{30}$$

and the ratio between the linear and simplicial representation errors, namely, $\rho \triangleq \sigma_{Lin}/\sigma_{Simp}$, is given by

$$\rho = \frac{\sqrt{12N}}{2k} 2^{-r+p}. \tag{31}$$

Table 1 shows the value of ρ for equal number of parameters ($r = p$) and several number of inputs. For increasing number of inputs (always considering $N < Q$), the accuracy of the simplicial expression improves with $\sqrt{N}$ over a standard VVM.

On the other hand, if the number of inputs is large, that is, $N > > Q$, then $T = Q$ and the standard deviation is

$$\sigma(\delta y) \approx \frac{2kR_wR_x}{\sqrt{12}^3} \sqrt{\frac{N}{Q}} \times 2^{-p} = \frac{2kR_wR_x}{\sqrt{12}^3} \sqrt{N} \times 2^{-p-q/2}. \tag{32}$$

The relative standard deviation is given by

$$\frac{\sigma(\delta y)}{R_y} \approx \frac{1}{\sqrt{12}} \times 2^{-p-q/2}, \tag{33}$$

and the representation error ratio by

$$\rho = \frac{\sqrt{12}}{2k} 2^{-r+p} 2^{q/2}. \tag{34}$$

Table 2 shows the value of ρ for equal number of parameters ($r = p$) and several values of input precision q. The accuracy of the simplicial expression improves exponentially with respect to the precision of the input in bits q (or what is equivalent, proportionally to $\sqrt{Q}$), over a standard VVM.

3.2 | Input quantization

This section analyzes the expression of the resulting errors when only the inputs are quantized. If every component of the inputs is quantized with q bits, then x_j can be written as $x_j = \hat{x}_j + \delta x_j$. In the VVM, the variance of the error is

$$var(\delta y) = var\left(\sum_{i=1}^{N} \hat{w}_i \delta x_i\right) = \sum_{i=1}^{N} var(\hat{w}_i) var(\delta x_i) = \frac{R_w^2 N}{12} \frac{(R_x 2^{-q})^2}{12} = \frac{(R_w R_x)^2 N}{12^2} 2^{-2q}. \tag{35}$$

Accordingly, the standard deviation is

TABLE 1 Ratio between linear and simplicial representation errors for different values of N ($p = r$) in the small number of inputs case

N	4	16	64	256	1024
	1.15	2.3	4.6	9.2	18.5

TABLE 2 Ratio between linear and simplicial representation errors for different values of q ($p = r$) in the large number of inputs case

q	4	5	6	7	8	9	10
	2.3	3.2	4.6	6.5	9.2	13	18.5

$$\sigma(\delta y) = \frac{R_w R_x \sqrt{N}}{12} 2^{-q}, \tag{36}$$

and considering the expression for the output range (20), the relative standard deviation is

$$\boxed{\frac{\sigma(\delta y)}{R(y)}_{Lin} \approx \frac{1}{2k} \times 2^{-q}}. \tag{37}$$

In the simplicial case, the output satisfies $y + \delta y = \langle \mathbf{w}, \hat{\mathbf{x}} + \delta \mathbf{x} \rangle = \langle \mathbf{c}, \hat{\mu} + \delta \mu \rangle$. In addition, $y = \langle \mathbf{c}, \hat{\mu} \rangle = \langle \mathbf{c}, D\hat{\mathbf{x}} \rangle = \langle D^T \mathbf{c}, \hat{\mathbf{x}} \rangle = \langle \mathbf{w}, \hat{\mathbf{x}} \rangle$, so taking differences we obtain

$$\delta y = \langle \mathbf{w}, \delta \mathbf{x} \rangle = \langle \mathbf{c}, \delta \mu \rangle. \tag{38}$$

Accordingly, the variance of the error, namely,

$$var(\delta y) = var\left(\sum_{j=1}^{N+1} \delta \mu_j c_j \right) \tag{39}$$

is exactly the same as in the liner case (36); therefore,

$$\boxed{\frac{\sigma(\delta y)}{R(y)}_{Simp} \approx \frac{1}{2k} \times 2^{-q}}. \tag{40}$$

3.3 | Total quantization

This section summarizes the expression of the resulting errors when both inputs and parameters are quantized.

In the VVM, the variance of the error is

$$\begin{aligned} var(\delta y) &= var\left(\sum_{i=1}^{N} \delta w_i \hat{x}_i + \hat{w}_i \delta x_i \right) \\ &= \frac{(R_w 2^{-r})^2}{12} \frac{R_x^2 N}{12} + \sum_{i=1}^{N} var(\hat{w}_i) var(\delta x_i) \\ &= \frac{(R_w 2^{-r})^2}{12} \frac{R_x^2 N}{12} + \frac{(R_x 2^{-q})^2}{12} \frac{R_w^2 N}{12} \\ &= \frac{(R_w R_x)^2 N}{12^2} \left(2^{-2r} + 2^{-2q} \right). \end{aligned} \tag{41}$$

Accordingly, the standard deviation is

$$\sigma(\delta y) = \frac{R_w R_x \sqrt{N}}{12} \sqrt{2^{-2r} + 2^{-2q}}. \tag{42}$$

Considering approximation (20) for the output range, the standard deviation relative to the output range can be approximated as follows:

$$\boxed{\frac{\sigma(\delta y)}{R(y)}_{Lin} \approx \frac{1}{2k} \sqrt{2^{-2r} + 2^{-2q}}} \tag{43}$$

In the simplicial case, the variance of the error is

$$var(\delta y) = var\left(\sum_{j=1}^{N+1} \hat{\mu}_j \delta c_j\right) + var\left(\sum_{j=1}^{N+1} \delta\mu_j \hat{c}_j\right) = \frac{1}{(2k)^2} R_c^2 2^{-2q} + \frac{1}{12T} R_c^2 2^{-2p} \tag{44}$$

The relative standard deviation can be written as

$$\boxed{\frac{\sigma(\delta y)}{R(y)}_{Simp} = \sqrt{A(q) + B(p,t)}} \tag{45}$$

where $A(q) = 1/(2k)^2 \times 2^{-2q}$, $B(p,t) = (1/12)2^{-2p-t}$ and $t \triangleq \log_2(T)$.

Example 3.2. A numerical simulation was performed using random values of inputs and parameters for different values of $N = 16, 64, 256, 1024, 4096$. Inputs and parameters were uniformly distributed in the interval $[-1, 1]$, sweeping the parameter accuracy (p, r) from 3 to 12 bits and the input accuracy q from 3 to 12 bits. For every combination of number of inputs and parameter/input bits, 10,000 random values were evaluated.

Figure 8 shows the output error (relative standard deviation) as a function of parameter quantization, with p and r between 3 and 12 bits, keeping the input quantization constant at $q = 6$ bits. The simplicial expression shows a lower error than the VVM for lower values of parameter quantization. This effect is more pronounced for higher number of inputs. Eventually, for sufficiently large number of bits in the parameters $(p, r > q)$, both expressions achieve the same accuracy. The numerical data shows good agreement with the theoretical expression.

Figure 9 shows the output error (relative standard deviation) as a function of input quantization, with q between 3 and 12 bits, keeping constant the parameter quantization at $p, r = 6$ bits. The VVM reaches a plateau after $q = 6$. The simplicial expression also reaches a plateau but at a higher value of q and with a smaller value. The effect is more pronounced the higher the number of inputs is. The numerical data show good agreement with the theoretical expression.

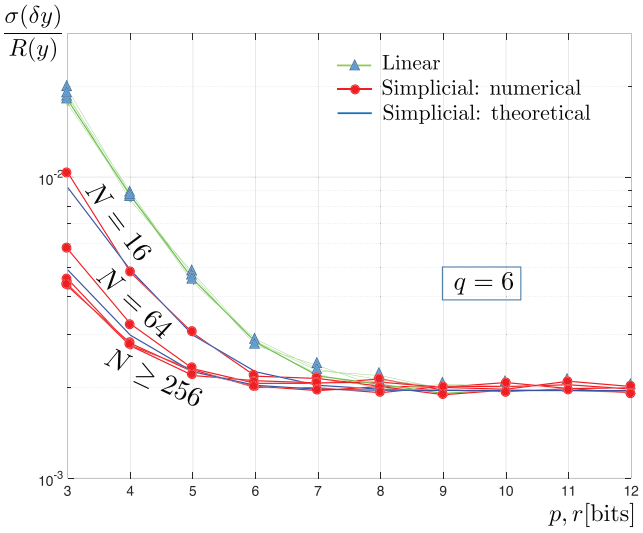

FIGURE 8 Numerical simulation of the relative standard deviation of the linear function, the simplicial function, and its theoretical value from Equation (44), for several values of $N = 16, 64, 256, 1024, 4096$, as a function of parameter quantization. Input quantization is $q = 6$ bits

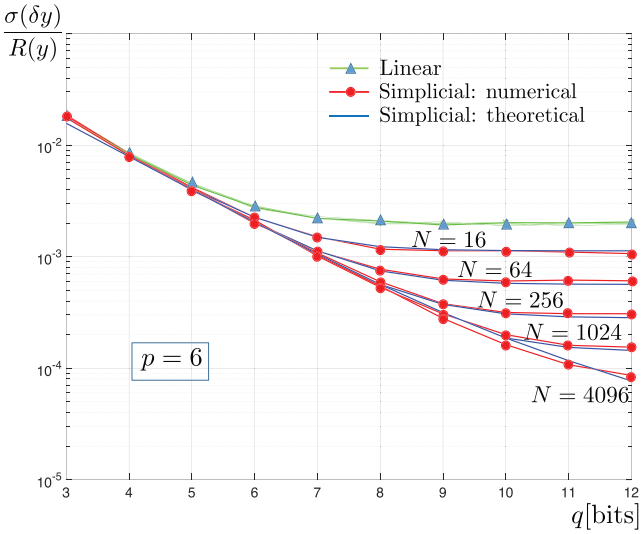

FIGURE 9 Numerical simulation of the relative standard deviation of the linear function, the simplicial function, and its theoretical value from Equation (44), for several values of $N = 16, 64, 256, 1024, 4096$, as a function of input quantization. Parameter quantization is $p = r = 6$ bits

3.4 | Choice of parameter precision

Figure 10 illustrates the error relative standard deviation for the VVM (solid-triangles) and the simplicial expression (solid-circles) as a function of p and r with a constant input precision $q = 8$.

In the linear case, once $r \geq q$ the error decreases marginally, therefore the optimal value for r is $r^{\diamond} = q$. In the example of Figure 10, this implies $r^{\diamond} = 8$.

In the simplicial case, the term $A(q) = 1/(2k)^2 \times 2^{-2q}$ is constant, since q is fixed and equal to 8; $\sqrt{A(q)}$ is indicated in solid line in Figure 10.

The second term, $B(p,t) = (1/12)2^{-2p-t}$ is a decreasing function of p and equals $B(p,q) = (1/12)2^{-2p-n}$, for $n \leq q$, where $n \triangleq \log_2(N)$. As the value of n increases, this term reduces by a factor 2^{-n}. For $n \geq q$, $t = q$ and $B(p,t)$ remain unchanged, since it is no longer a function of n but of q, that is, $B(p,t) = B(p,q)$. The term $\sqrt{B(p,t)}$ is plotted with a dot-dash line in Figure 10 for two different values of n($n = 6$ and $n \geq 8$, corresponding to $N = 64$ and $N = 256$ inputs, respectively).

For the large number of inputs case, $n \geq q$, the intersection of both terms occurs when $A(q) = B(p,q)$:

$$\frac{1}{(2k)^2} 2^{-2q} = \frac{1}{12} 2^{-2p^{\diamond} - q}. \tag{46}$$

After some algebraic manipulation, the solution is

$$p^{\diamond} = \frac{q}{2} + \frac{1}{2}\log_2(3). \tag{47}$$

For example, if the input precision is $q = 8$ bits and there are more than $N = 256$ inputs, corresponding to $n = 8$, the optimum is $p^{\diamond} = 4.8$ bits, a value remarkably close to the estimated 4.7 bits of information for hippocampal synapses.[27] In the VVM, when $q = 8$ bits, the optimum parameter precision is $r^{\diamond} = 8$ regardless of the number of inputs. In this case, both representations use the same precision q for the input and produce the same error, but the simplicial expression does so using less precision for the parameters. In fact, while the linear expression needs $r = q$, the simplicial needs $p \approx q/2 + 0.8$.

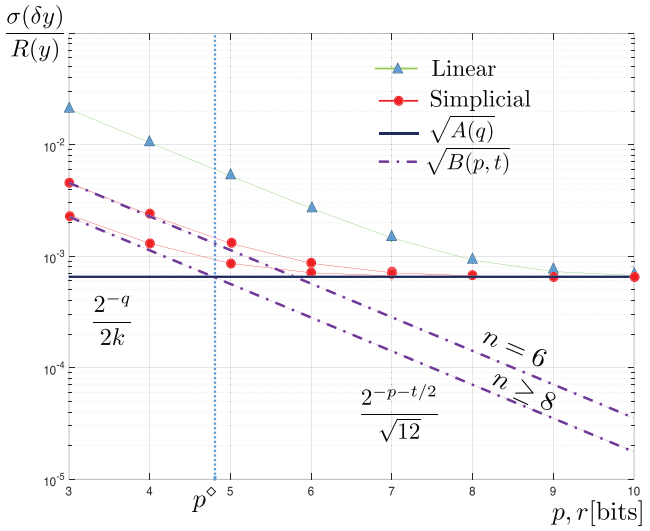

FIGURE 10 Relative standard deviation of the error with respect to the parameters precision: linear expression of Equation (43), simplicial representation for $N = 64, 1024$, and the two individual terms in Equation (45). Input quantization is $q = 8$ bits

4 | COMPUTATION ENERGY

In modern technologies, most of the energy consumed by a digital circuit is dynamic, that is, is the consequence of the charge and discharge of MOS transistor capacitances.[28] Accordingly, the total energy demanded by each implementation will be proportional (by a technology dependent constant) to the number of full/half adder (FA/HA) circuits and the number of times they switch (activity) during the period of time needed to complete the operation. Of course, in a VLSI implementation, there is circuitry like registers and multiplexers that will also be necessary, but will be present in both cases.

In the VVM, a parallel linear vector–vector product between an N r-bit word and an N q-bit word requires $N \times rq$ AND gates, $N \times r$ HA and $N \times (q-1)r$ FA, followed by an adder of $N(r+q)$-bit inputs, which requires $(N-1)$ HA and $(N-1)(r+q) - \log(N)$ FA. The total number of gates, considering two gates per HA and five per FA, is

$$GA_{Lin} = N \times rq + 2N \times r + 5N \times (q-1)r + 2(N-1) + 5((N-1)(r+q) - \log(N)). \tag{48}$$

The energy is the product of (48) times 1 cycle, so it is the same expression as (48)

$$E_{Lin} = N \times rq + 2N \times r + 5N \times (q-1)r + 2(N-1) + 5((N-1)(r+q) - \log(N)). \tag{49}$$

The linear VVM can also be implemented in a serial fashion, which requires fewer gates (one multiplier and an accumulator) and more execution cycles; however, in both cases the required energy is similar.

In the simplicial, case it is necessary to add Q numbers of p-bits. This requires $(Q-1)p - \log(Q)$ FA and $Q-1$ HA, and assuming that this is accomplished in one cycle, the energy (and also the number of gates) is given, in principle, by

$$E_{Simp} = 2(Q-1) + 5((Q-1)p - \log(Q)). \tag{50}$$

In addition, the simplicial representation requires a prior sorting operation to generate the μ values. This operation can be done in two possible ways. The first option is sorting the inputs and taking the differences, as implemented in Agustin Rodriguez et al.[29] This requires $\mathcal{O}(N\log_2(N))$ comparisons of q bits and then N subtractions of q bits. The second option that turns out to be much more efficient from the energy/area viewpoint (extensively used in compact realizations like other studies[30,3120]) consists of encoding inputs in time by comparing them with an increasing ramp signal, as illustrated in Figure 11A. The ramp always has Q clock cycles, regardless of the number of inputs N. In this case, the sorting is naturally obtained as the time length of each encoded signal, and the μ values are directly the intervals of time

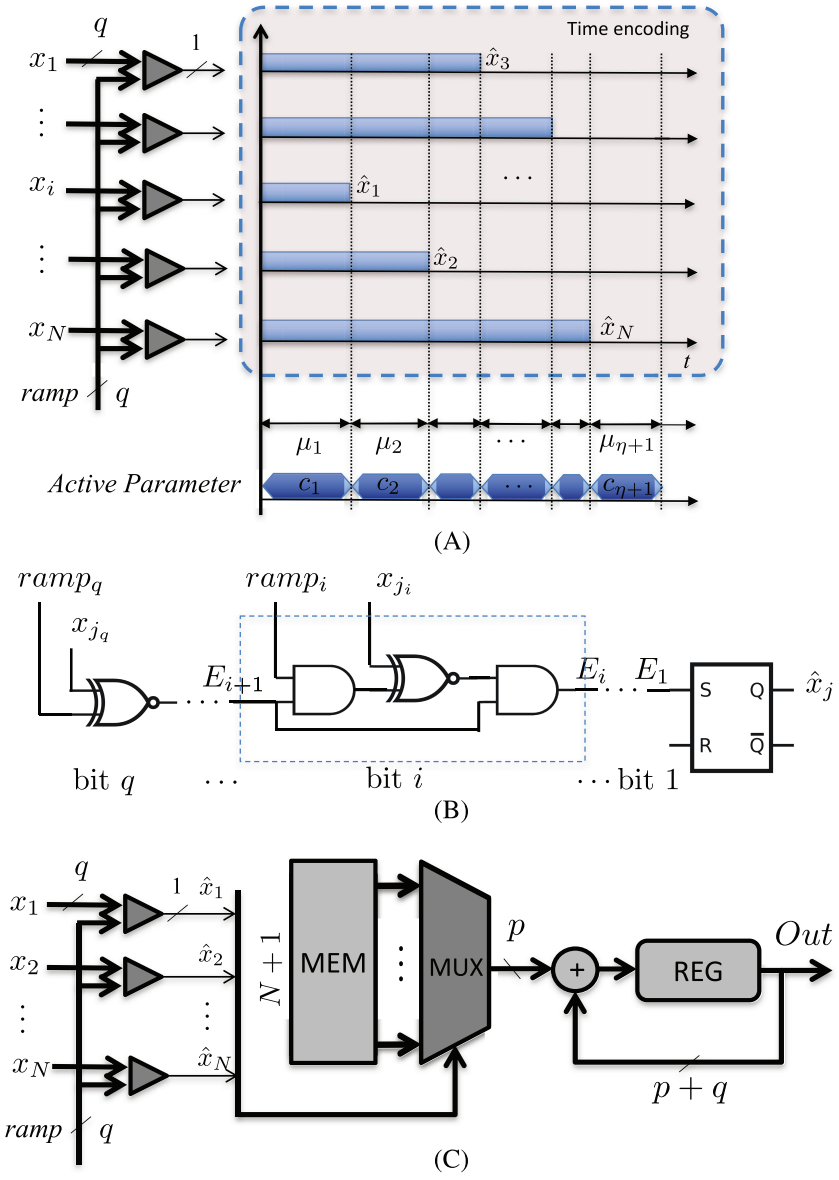

FIGURE 11 (A) Intrinsic sorting obtained by encoding inputs in time; (B) efficient time-encoding circuit block corresponding to the *j*th input; (C) architecture to compute in time

between changes in the inputs. These intervals can be used to produce summation (7) in time, without explicit multiplications.

The comparator can be efficiently implemented with N parallel blocks like the one illustrated in Figure 11B. The comparison starts by the most significant bit; whenever a coincidence is detected, the next block to the right (the following most significant bit) is enabled to compare. This block only produces an enable output to the following block when it is enabled, *and* there is a coincidence between the ramp and the corresponding input bit. The last block in the cascade corresponds to the least significant bit; when the enable output of this block becomes active, then the equality of the whole input has been detected and the SR latch is activated. It is easy to see that the gates in every block of the cascade are only activated once during a ramp cycle. Therefore, the energy spent by the entire cascade during a ramp cycle, including the SR latch, is

$$E_C = 1 + 3(q-1) + 2 = 3q. \tag{51}$$

The entire comparator energy, considering N inputs, is given by

$$E_C = 3Nq. \tag{52}$$

The $\hat{x}_j$ values form a word that addresses the parameter memory and outputs the parameters sequentially. These parameters can be added serially, using standard digital circuitry as illustrated in Figure 11C. This is done with an accumulator composed by an adder with $p-1$ FA and $q+1$ HA, which implies a total number of gates:

$$GA_{Accum} = 2(q+1) + 5(p-1) \tag{53}$$

in Q cycles. In this case, the energy required is

$$E_{Simp} = Q(2(q+1) + 5(p-1)). \tag{54}$$

In summary, the energy demanded by the simplicial algorithm is

$$\mathcal{O}(E_{Simp}) = Q(p+q) + Nq. \tag{55}$$

The energy required by the VVM is $\mathcal{O}(E_{Lin}) = Nqr$, so it depends on the product of the number of inputs and the representation bits of inputs and parameters. In the simplicial case, the computation energy has a complexity $\mathcal{O}(E_{Comp}) = Q(p+q)$, independent of the number of inputs, while sorting has a complexity $\mathcal{O}(E_{Sorting}) = Nq$ that depends on the number of inputs and their representation bits, but not on the parameters. This favors the simplicial representation when $N >> Q$, as illustrated in Figure 12A.

In the case of a matrix–vector multiplication where multiple outputs need to be calculated for the same input, for example, when

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} \tag{56}$$

with $\mathbf{y} \in \mathbb{R}^M$, the cost of sorting is constant and independent of the number of outputs M. This situation is typical in neural networks where every layer implements a number M of neurons using the previous layer output as input (and also in convolutional kernels where M kernels are used for any given input set). In this case, the energy demanded by the simplicial and VVM representations are respectively

$$\mathcal{O}(E_{Simp}) = Nq + MQ(p+q) \tag{57}$$

$$\mathcal{O}(E_{Lin}) = M(Nqr + Nr + Nq). \tag{58}$$

The energy in the VVM is the product of the number of input times the number of outputs, while in the simplicial case it is decoupled. This produces a significant saving in computing energy, especially for large N. Figure 12B illustrates the situation for a case where $M = 1000$ different outputs need to be computed.

Lastly and for the sake of completeness, Figure 13 shows the number of gates (area) required by each implementation for several inputs, with $q = 8$ and parameters $p, r = 2, 4, ..., 12$. The serial versions are much more compact than their parallel counterparts. Notice that in the serial simplicial case, most of the gates correspond to the sorting operation. As previously mentioned, in the case of a MVM, the sorting units are shared, which reduces the overall area.

5 | APPLICATION EXAMPLE

In order to illustrate the advantages of the proposed methodology, we present in the following examples the implementation of a deep morphological network (DMN) using the proposed simplicial architecture for the recognition of city maps.[32] As previously mentioned, morphological operations are a concatenation of min and max operations on the inputs. In particular, DMN apply morphology operations on weighted inputs;[33] thus, they are a special case of weighted OS filters and can be considered as functions defined over a single simplex as in (1). In particular, we consider the DMN architecture proposed in Nogueira et al,[26] which is illustrated in Figure 14.

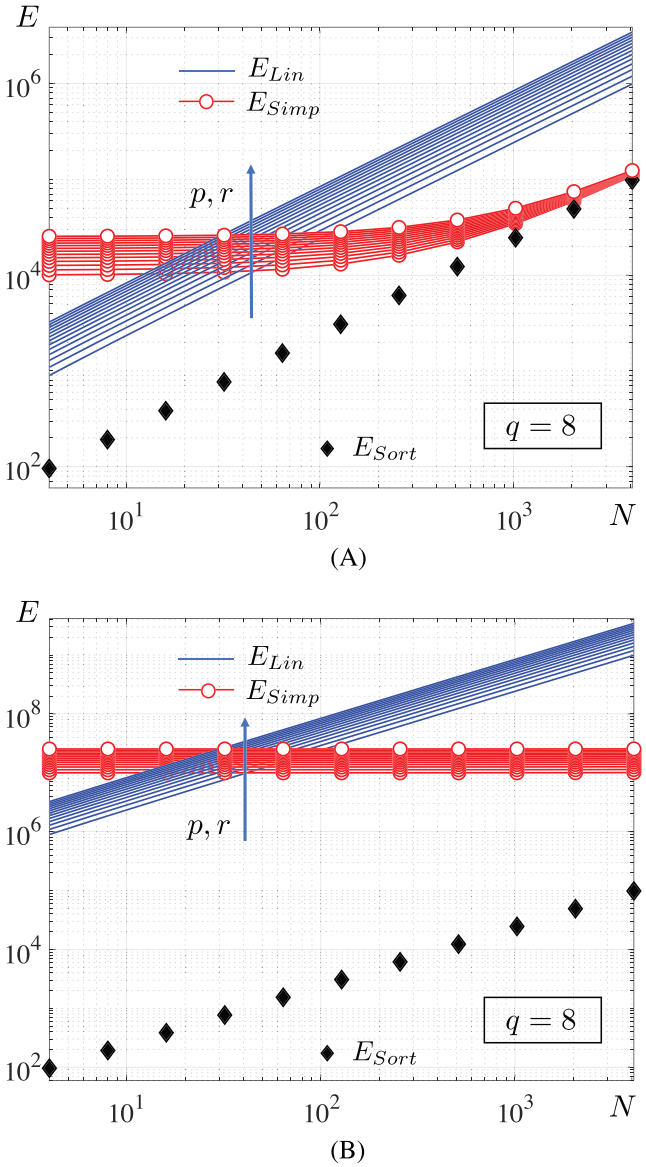

FIGURE 12 Energy for the lineal expression (49) and the simplicial expression (54) as a function of N for $p, r = 2, 4, \ldots, 16$: (A) $q = 8$; (B) $q = 8$ and $M = 1000$

Example 5.1. The DMN illustrated in Figure 14 has one morphological layer with six 11×11 input kernels, a six-channel 2×2 MaxPool layer, three fully connected (FC) layers with rectified linear units (ReLUs), and a final SoftMax classification layer. In our proposed architecture, we replace the first layer with simplicial units which operate on $11 \times 11 = 121$ inputs.

We trained the network using the UC Merced Land Use Dataset during 800 epochs, by stochastic gradient descent (SGD) method. All images were reduced to 224×224 pixels and were split into 1260 images for training, 420 for validation and 420 to test the net performance. We used a batch size of four samples and a learn rate of 0.01 for the simplicial coefficients, linear weights, and biases, with an exponentially decay factor of 0.99, reaching 95.79% accuracy on the training set, 82.14% on the validation set and 80.95% on the test set. As a comparison, Nogueira et al[26] report 76.7% accuracy.

In order to assess the appropriate level of quantization for a digital implementation, we quantized the parameters of the first layer (simplicial function only), for different input quantization levels. We computed the output of the simplicial layer for all the images used and subtracted it from the full precision simplicial output. We divided this difference by the range of the original output, applied norm 2 and divided by the

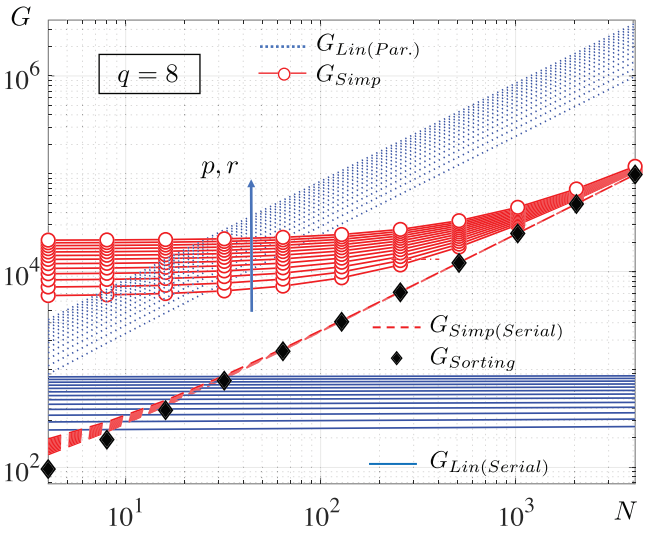

FIGURE 13 Gates required by the simplicial (50) and serial simplicial (53) interpolation, parallel and serial linear VVM, and sorting operation for $q = 8$ and $p, r = 2, 4, ..., 12$ as a function of N

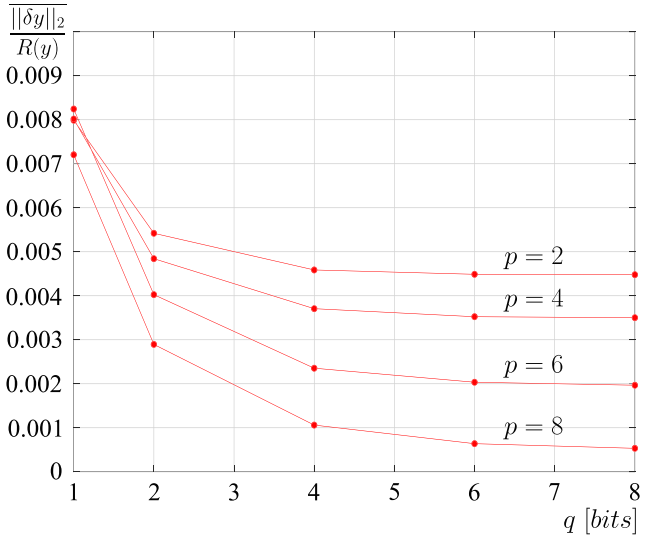

FIGURE 14 DMN architecture

number of output neurons. This error value was then averaged for each image, each input channel, and each simplicial filter, resulting in the plot of Figure 15. For input quantization levels of more than 4 bits, the accuracy does not change significantly. This is confirmed by the plot of classification accuracy versus input/parameter quantization of Figure 16.

For the second part of the example, we designed a digital core to implement the simplicial DMN and a standard MAC implementation based on the state-of-the-art architecture presented in Gokhale et al.[34] Both modules are parametrized to achieve the same throughput in the context of a pipelined system.

Example 5.2. We designed two modular digital blocks with 128 inputs, so that they can handle the 11×11 kernel operations. Based on Figure 16, we selected an input precision $q = 4$ for both cores. In the simplicial case, we chose $p = 3$ bits for the parameters and in the standard linear, $p = 4$, in order to achieve the same precision, in accordance with the results of Section 3.

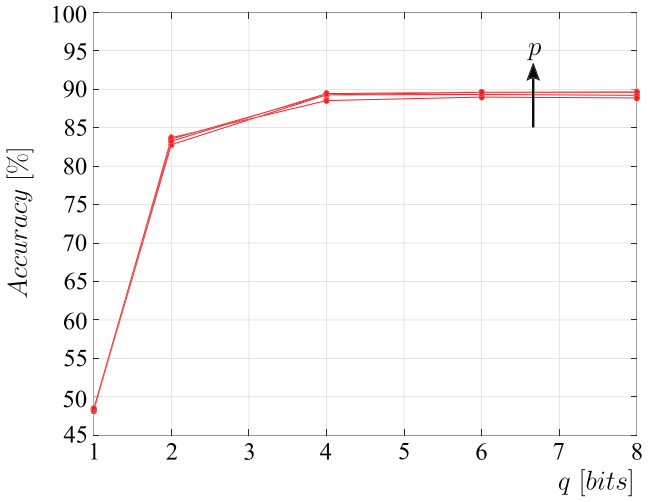

FIGURE 15 Simplicial layer error as a function of the input quantization

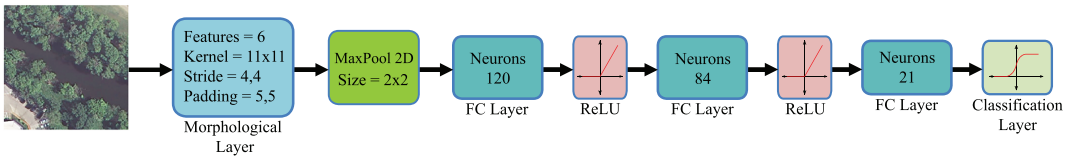

FIGURE 16 Classification accuracy of the training set as a function of the input quantization

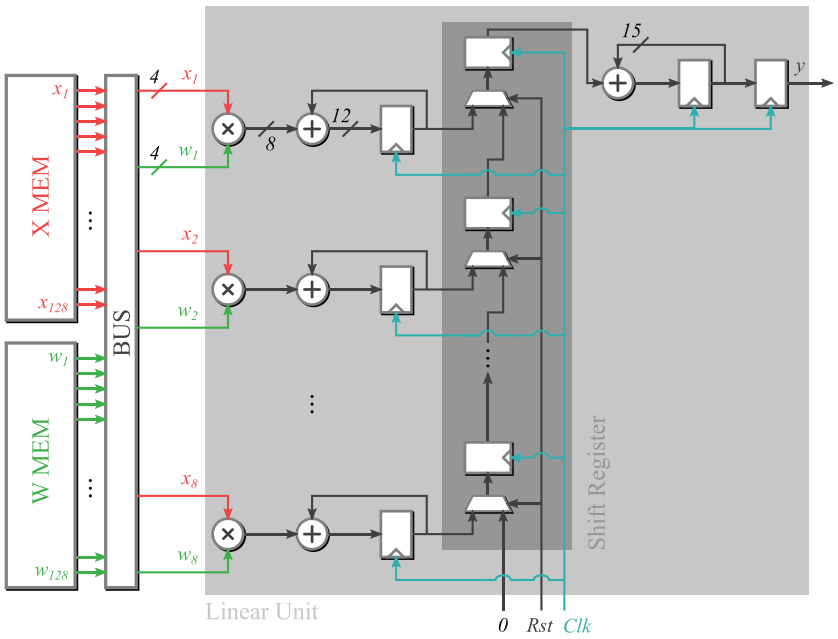

FIGURE 17 Synthesized architecture for the linear block

The standard linear architecture is based on the state-of-the-art implementation[34] illustrated in Figure 17. The first stage consists of eight channels that multiply and accumulate 16 inputs each with their corresponding coefficients, one at a time, in 16 clock cycles. The second pipeline stage, takes the eight

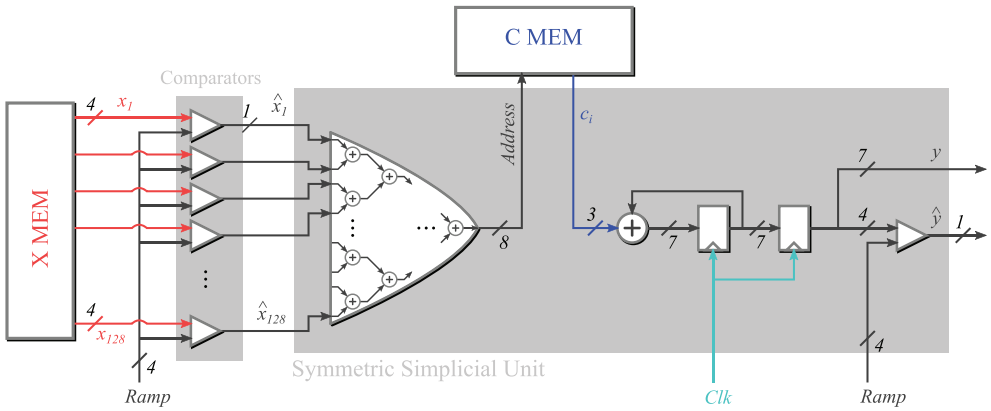

FIGURE 18 Synthesized architecture for the simplicial block

TABLE 3 Synthesis results of the simplicial and a standard linear 128 4-bit input core in a 65-nm technology, running at 50 MHz with an operation voltage of 1.08 V

	Simplicial	Linear
Combinational cells	175	828
Sequential cells	17	210
Buff/Inv	10	120
Total area	1130.4 μm^2	3787 μm^2
Dyn. power	17.46 μW	182.27 μW
Leakage power	30.80 nW	141.22 nW
Total power	17.5 μW	182.4 μW

partial results and adds them serially in eight clock cycles. Both stages work in pipeline, so the total computing time is 16 clock cycles.

The simplicial architecture is shown in Figure 18 and follows the principle previously described in Figure 11C. It is composed of 128 4-bit inputs that are time-encoded with 1-bit lines during a ramp period of 16 clock cycles. They enter an adder tree, which generates the addresses of the coefficients that are subsequently added in an accumulator. This architecture also computes in 16 clock cycles.

Both architectures were described in SystemVerilog HDL language and synthesized with Synopsys® DesignCompiler on a 65nm technology for an operation frequency of 50 MHz at 1.08 V. In both cases, the designs were optimized using clock-gating cells to reduce area and switching activity. Post-synthesis simulation and power analysis were performed with Siemens® QuestaSim and Synopsys® PowerCompiler, respectively. Table 3 summarizes the results. The simplicial block is three times smaller and consumes 10 times less power for the same processing time.

6 | CONCLUSIONS

We have shown that the simplicial interpolation is a convenient alternative to implement VVM, from accuracy, storage, and energy viewpoints. Actually, the advantage is more pronounced for large number of inputs, which is relevant given the current interest in large-scale computation demanded by machine learning and artificial intelligence architectures. The simplicial representation requires an encoding of the inputs, which need to be sorted first and then subsequently subtracted. When the number of inputs is large, the number of nonzero encoded inputs is at most equal to the number of levels of the input, namely $Q = 2^q$, and the variance is small $1/Q^2$. As a consequence, the parametric quantization error has a reduced impact on the output error. In this situation, the simplicial representation achieves the same error

as a standard VVM with approximately half the number of parameter bits. This translates directly into a saving in parameter storage and also has an impact on energy, as explained next.

Depending on the number of inputs, the simplicial architecture exhibits better results either on area or energy. For moderate and large number of inputs (Figure 12), the simplicial expression achieves better energy performance but a serial VVM achieves a more compact implementation (Figure 13). For small number of inputs, the situation is the opposite.

Most of the area of the simplicial approach is due to the encoding comparators regardless the number of inputs. In terms of energy, the comparators become the predominant factor for very large number of inputs. We have shown that if the same inputs need to be re-used by different computational blocks, the encoding operation can be shared and the corresponding area/energy budget is significantly reduced. This is indeed the case for neural networks structures where every neuron of a layer feeds all the neurons of the next layer. This can be exploited in multilayer neural networks, especially if they are very large, by an adequate architecture design to produce more area/energy efficient cores.

Finally, it is worth noting that the accuracy properties are inherent to the nature of the computation, namely, input encoding and interpolation. This is true whether we consider a single domain simplex (as in the case of OS filters) or a more general region, where a supervisory system could be in charge of detecting different regions of the domain and assigning the corresponding parameters.

ACKNOWLEDGMENTS

This work was partially supported by the following grants: (PICT 2016 No. 2009, PICT 2017 No. 2526) ANPCyT of Argentina and (PGI 2019 No. 26/K086) UNS.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID


P. Julian https://orcid.org/0000-0002-6308-4497

REFERENCES

1. Karakiewicz R, Genov R, Cauwenberghs G.. 1.1 TMACS/mW fine-grained stochastic resonant charge-recycling array processor. *Sensors J IEEE*. 2012;12(4):785-792.
2. Lin TY, Payne AJ. Programmable analogue vector–matrix multiplier. *Electron Lett*. 2002;38(1):1-2.
3. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger K. Q., eds. *Advances in Neural Information Processing Systems 25*. New York: Curran Associates, Inc.; 2012:1097-1105.
4. Sim J, Park JS, Kim M, Bae D, Choi Y, Kim LS. 14.6 A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems. In: 2016 IEEE International Solid-State Circuits Conference (ISSCC); San Francisco, CA, USA; 2016:264-265.
5. Moons B, Uytterhoeven R, Dehaene W. 14.5 Envision: 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28-nm FDSOI. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC). IEEE; 2017:246-247.
6. Chen YH, Krishna T, Emer JS, Sze V. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J Solid-State Circ*. 2017;52(1):127-138.
7. Mounica C, Krishna S, Veeramachaneni S, Mahammad N. Efficient implementation of mixed-precision multiply-accumulator unit for AI algorithms. *Int J Circuit Theory Appl*. 2020;48(8):1386-1394.
8. Zhou A, Yao A, Guo Y, Xu L, Chen Y.. Incremental network quantization: towards lossless CNNs with low-precision weights. arxiv.org; 2017.
9. Jung S, Son C, Lee S, et al. Learning to quantize deep networks by optimizing quantization intervals with task loss. arXiv.org; 2018.
10. Park H, Kim D, Kim S. TMA: Tera-MACs/W neural hardware inference accelerator with a multiplier-less massive parallel processor. *Int J Circuit Theory Appl*. 2021;49(5):1399-1409.
11. Wu J, Leng C, Wang Y, Hu Q, Cheng J. Quantized convolutional neural networks for mobile devices. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016; Las Vegas, NV, USA:4820-4828.
12. Han S, Liu X, Mao H, Pu J, IEEE APA. EIE: efficient inference engine on compressed deep neural network. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. Vol. 1. Piscataway, NJ, USA: IEEE; 2016:243-254.
13. Garland J, Gregg D.. Low complexity multiply–accumulate units for convolutional neural networks with weight-sharing. *ACM Trans Architect Code Optim (TACO)*. 2018;15(3):31-24.
14. Rastegari M, Ordonez V, Redmon J, Farhadi A. XNOR-Net: ImageNet classification using binary convolutional neural networks. arXiv.org; 2016.

15. Wu B, Wang Y, Zhang P, Tian Y, Vajda P, Keutzer K. Mixed precision quantization of ConvNets via differentiable neural architecture search. arXiv.org; 2018.
16. Chien MJ, Kuh ES. Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision. *IEEE Trans Circ Syst*. 1977;24(6):305-317.
17. Julian P, Desages A, Agamennoni O.. High-level canonical piecewise linear representation using a simplicial partition. *IEEE Trans Circ Syst I: Fundamental Theory Appl*. 1999;46(4):463-480.
18. Lum R, Chua LO. How to represent continuous piecewise linear functions in closed form. *Int J Circuit Theory Appl*. 2006;34(6):617-635.
19. Julian P, Desages A, Agamennoni O. High-level canonical piecewise linear representation using a simplicial partition. *IEEE Trans Circ Syst I: Fundamental Theory Appl*. 1999;46(4):463-480.
20. Di Federico M, Julian P, Mandolesi PS. SCDVP: a simplicial CNN digital visual processor. *IEEE Trans Circ Syst I: Regular Papers*. 2014; 61(7):1962-1969.
21. Villemur M, Julian P, Andreou AG. Energy aware simplicial processor for embedded morphological visual processing in intelligent Internet of things. *Electron Lett*. 2018;54(7):420-422.
22. Pitas I, Venetsanopoulos AN. Nonlinear order statistic filters for image filtering and edge detection. *Signal Process*. 1986;10(4):395-413.
23. Barner KE, Arce GR. *21 Order-statistic filtering and smoothing of time-series: part II*, Vol. 17; 1998;555-602.
24. Lucat L, Siohan P, Barba D.. Adaptive and global optimization methods for weighted vector median filters. *Signal Process: Image Commun*. 2002;17(7):509-524.
25. Maragos P, Schafer R, International RSII. A unification of linear, median, order-statistics and morphological filters under mathematical morphology. *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 10. Tampa, FL, USA: IEEE; 1985:1329-1332.
26. Nogueira K, Chanussot J, Mura MD, Schwartz WR, Santos dJA. An Introduction to deep morphological networks; 2019.
27. Bartol TM, Bromer C, Kinney J, Chirillo MA, Bourne JN. Nanoconnectomic upper bound on the variability of synaptic plasticity, 4; 2016.
28. Alioto M. *Enabling the Internet of Things*. USA: Springer; 2017.
29. Agustin Rodriguez J, Lifschitz OD, Manuel Jimenez-Fernandez V, Julian P, Enrique Agamennoni O. Application-specific processor for piecewise linear functions computation. *IEEE Trans Circ Syst I: Reg Papers*. 2011;58(5):971-981.
30. Julian P, Dogaru R, Chua L. A piecewise-linear simplicial coupling cell for CNN gray-level image processing. *IEEE Trans Circ Syst I: Fundamental Theory Appl*. 2002;49(7):904-913.
31. Mandolesi PS, Julian P, Andreou AG. A scalable and programmable simplicial CNN digital pixel processor architecture. *IEEE Trans Circ Syst I: Regular Papers*. 2004;51(5):988-996.
32. Yang Y & Newsam S Bag-of-visual-words and spatial extensions for land-use classification. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems; San José, CA, USA; 2010:270-279.
33. Shen Y, Zhong X, Shih FY. Deep morphological neural networks. arXiv e-prints 2019: arXiv:1909.01532.
34. Gokhale V, Zaidy A, Chang AXM, Culurciello E. Snowflake: a model agnostic accelerator for deep convolutional neural networks. arXiv.org; 2017.

How to cite this article: Julian P, Andreou AG, Villemur M, Rodriguez N. Simplicial computation: A methodology to compute vector–vector multiplications with reduced complexity. *Int J Circ Theor Appl*. 2021;49 (11):3766-3788. doi:10.1002/cta.3128