IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 19, 2022, accepted May 31, 2022, date of publication June 13, 2022, date of current version June 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182719

ReSNN-DCT: Methodology for Reduction of the Spiking Neural Network Using Discrete Cosine Transform and Elegant Pairing

FRANCISCO DE ASSIS PEREIRA JANUÁRIO[1] AND JOSÉ REGINALDO HUGH CARVALHO[2]
[1]Department of Electronics and Computing, Federal University of Amazonas, Manaus, Amazonas 69067-005, Brazil
[2]Institute of Computing, Federal University of Amazonas, Manaus, Amazonas 69067-005, Brazil

Corresponding author: Francisco De Assis Pereira Januário (franciscojanuario@ufam.edu.br)

This work was supported in part by Motorola through the IMPACT-Laboratory Research and Development Project of the Institute of Computing (ICOMP); and in part by the Department of Electronics and Computing (DTEC), Federal University of Amazonas (UFAM).

ABSTRACT In recent years, the use of artificial neural network applications to perform object classification and event prediction has increased, mainly from research about deep learning techniques running on hardware such as GPU and FPGA. The interest in the use of neural networks extends to embedded systems, due to the development of applications in smart mobile devices, such as cell phones, drones, autonomous cars and industrial robots. In this article, a methodology is proposed that allows to reduce a spiking neural network, applying the discrete cosine transform (DCT) and elegant pairing. The Izhikevich model was used as a basis for the architecture of the spiking neural network. The simulation results demonstrate the effectiveness of the methodology, showing the feasibility of reducing synapses, applying the DCT transform, and of reducing neurons from the intermediate layers, using the elegant pairing technique of the coefficients, and maintaining the accuracy of the response of the spiking neural network. The results also demonstrate the contribution of the proposed methodology to the scalability of the neural network, with the increase in the storage capacity of the coefficients of the SNN layers.

INDEX TERMS Spiking neural network (SNN), Izhikevich model, discrete cosine transform (DCT), elegant pairing.

I. INTRODUCTION

The human brain has the ability to recognize, in a fraction of a second, diverse patterns such as varied shapes of clouds, words on a computer screen, types of flowers and wild animals, as well as the ability to interpret temporal data to predict events such as variation seasonal levels of a river. From the biological capacity of the brain, artificial neural network (ANN) models have been developed, widely used to solve pattern recognition problems [1].

Artificial neural networks are the basis for the development of classifiers and predictors, which process relevant characteristics of the input data provided to the network, and which are extracted in a pre-processing prior to classification. This stage of pre-processing is necessary so that the data is ready to be processed in a machine, capable of, for example, recognizing images. Generally, it is conceptualized that the machine is taught to learn [1], [2].

Among the neural network architectures, the most used are the convolutional neural networks (CNN), which have multiple layers, designed to work with two-dimensional data, being used to extract and classify features from a raw data set. Their core is composed of two types of layers: the convolutional layer, responsible for detecting characteristics of the previous layer; and the pooling layer, responsible for uniting similar characteristics into a group. The deep CNN networks work with spatial and temporal information, characteristics present in images and videos. The CNN is particularly useful for pattern recognition applications that classify images. Figure 1 shows the architecture of a typical convolutional neural network [1]–[4]

Another type of neural network architecture that is still in its state of the art, but has been gaining more interest in research and applications, are spiking neural

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Lai.

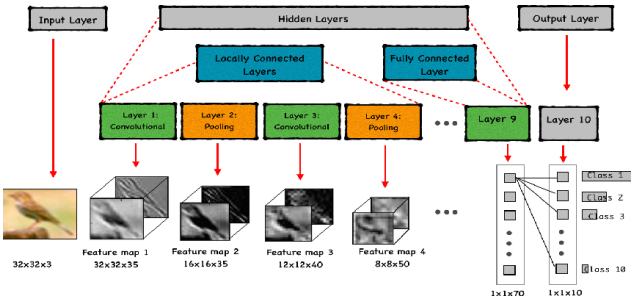

FIGURE 1. Architecture of a typical CNN network [1].

networks (SNN), which seek to mimic the functions of biological nervous systems, exploring new methods for cognitive computing. Its computational capacity, based on the spatial and temporal characteristics of the input signals, has proved to be very attractive for implementing energy-efficient hardware applications [5].

Neural networks have been used in the development of applications with large amount of raw data, mainly in the area of data mining, computer vision and pattern recognition, such as image recognition for medical diagnosis. However, processing a large dataset requires an increase in computational speed, otherwise the process becomes slow, due to the high computational cost of training neural networks, which may require the implementation of billions of neuron connections, such as in Google's cat videos recognition system. To solve this problem, computational parallelism in hardware is employed using clusters of graphics processing units (GPU) with general purpose graphic processors (GPGPU) [1], [2], [4], [6].

Three factors impact the use of GPU for the implementation of deep learning networks: the high cost per unit of graphics processing, the low flexibility to add more GPGPUs and the high power consumption. As an alternative, researchers use programmable hardware, such as field programmable gate arrays (FPGA), for the development of digital system designs. FPGA is a programmable logic device (PLD) of flexible architecture, encapsulated in a single integrated circuit, which provides high performance per watt of power consumption. Hardware accelerators of FPGA algorithms allow to implement complex structures with energy efficiency and high performance, besides providing a scalable architecture. For deep learning applications, FPGA is a suitable solution for development in hardware platforms. Its flexibility, low cost and parallel processing capability, allows classifying FPGA technology between general purpose processors (GPP) and application specific integrated circuits (ASIC) [3], [4], [6]–[8].

Highlighting the implementation in hardware, the interest in the use of neural networks extends to embedded systems, due to the development of applications in smart mobile devices, such as cell phones, drones, autonomous cars and industrial robots. However, in embedded systems, hardware limitations, such as memory and energy consumption, must be observed, which significantly impact the processing of a neural network. When using hardware to implement neural networks, there are some limitations that impact application performance, some of which are energy efficiency and scalability.

Energy efficiency is one of the most researched characteristics in deep neural network designs on hardware with limited resources. Research shows that the power consumption of GPGPU is very high compared to CPU. However, GPGPU-based accelerators have been used intensively for CNN implementations. Alternatives such as the FPGA have advantages over CPUs and GPGPUs due to their features of massive parallelism and low power consumption, which make them attractive for deep neural network applications. However, FPGA has loss of energy efficiency with the addition of more processing units.

Scalability is the factor that determines the level of processing capacity of the hardware. This capacity is directly associated with the number of processing and memory elements used for each type of neural network architecture implemented. A neural network architecture is said to be scalable in hardware if it can add more processing cores, increasing the layers of the network. However, when all hardware resources are in use, it is necessary to use another hardware platforms with more resources. This is a dynamic implementation problem. Scalability directly impacts the energy efficiency of the hardware. Current research focuses on the development of models that apply techniques capable of increasing the level of scalability of hardware limited in resources.

This article contributes with a proposed methodology for the reduction spiking neural networks (SNN), using the discrete cosine transform (DCT) and the elegant pairing technique. The objective of the methodology is to reduce the number of synapses and neurons, resizing the architecture of the neural network, and thus allowing an SNN to be implemented in hardware with limited resources, such as IoT and FPGA sensor devices, but maintaining the same accuracy, if compared with an SNN without using the approach proposed in this article. This technique contributes to improving the scalability of the SNN, making it possible to expand the processing capacity with the increase of more neurons and layers in the network architecture, and later, reducing the new architecture using the proposed methodology of ReSNN-DCT.

The paper is divided as follows: section II presents concepts about spiking neural networks (SNN) and the Izhikevich model. Section III, describes the discrete cosine transform (DCT) and related works of DCT applied in SNN neural networks. Section IV describes the elegant pairing applied in the ReSNN-DCT. Section V presents the proposed methodology for the reduction of layers of spiking neural networks. Section VI presents the experimental results of validation of the methodology proposed in this work. Finally, section VII, presents the final considerations about the work.

II. SPIKING NEURAL NETWORK

Spiking neural networks (SNN) are the third generation of artificial neural networks, which more realistically simulate the biological neuron, making use of spikes as input and output information, as shown in Figure 2. This functional feature allows the spatial and temporal encoding of information, through pulses [9].

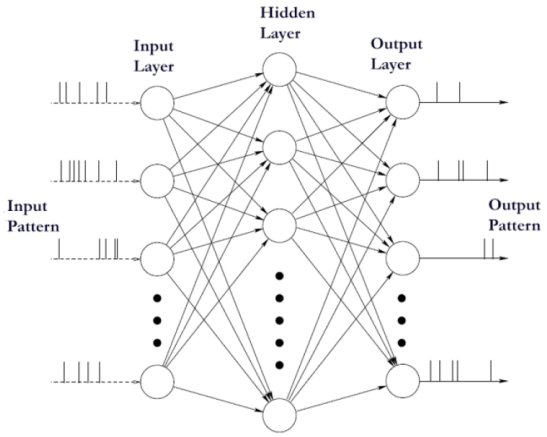

FIGURE 2. Architecture of a spiking neural network [10].

In a spiking neural network, signals consist of small electrical pulses called action potentials or spikes, and have an amplitude around 100 mV with a duration of $1-2$ ms. These neurons transmit sequences of spikes, called spike trains, that change dramatically in frequency over a short period of time, requiring spatial and temporal information from the input spike patterns to encode the message between neurons. Thus, the number and duration of spikes provide useful information for the network.

Mathematical models were developed to describe the behavior of a spiking neuron. Among these models, we can mention the Integrate-and-Fire model [11], proposed in 1907 by the neuroscientist Louis Lapicque, and the Hodgkin-Huxley model [12], presented in 1963 by Nobel Prize-winning physiologists and biophysicists Alan Hodgkin and Andrew Huxley, which provided an understanding of neuron behavior in terms of action potentials [13].

In a simplified model of a spiking neuron, a neuron v fires when there is, on the neuron's membrane, a potential value P_v greater than or equal to a threshold value θ_v. There are two types of neurons: postsynaptic potentials (EPSP) and presynaptic potentials (IPSP). The potential P_v is the sum of the outputs of neurons u, which are connected to neuron v through synapses. Whenever a presynaptic neuron u fires at time s, this results in a P_v at time t. Mathematically, this relationship is described by (1) in which $w_{u,v}$ is the weight, which represents the strength (effectiveness) between the synapes, and $\epsilon_{u,v}(t-s)$ is the response function that has the form illustrated in Figure 3 [13].

$$P_v = w_{u,v}\epsilon_{u,v}(t-s) \tag{1}$$

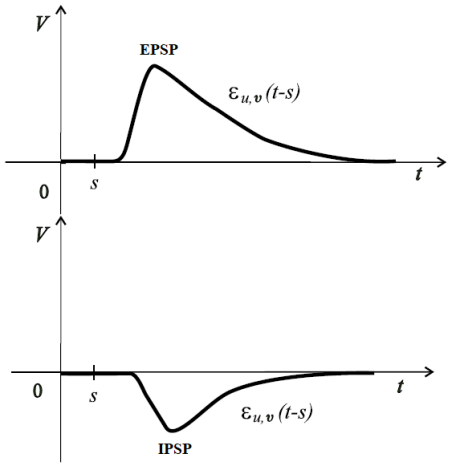

FIGURE 3. Typical form of response functions (EPSP and IPSP) of a biological neuron [13].

In a typical biological neuron, the resting membrane potential is approximately -70 mV, the firing threshold of a resting neuron is close to -50 mV. If a neuron v fires in time t', it will not fire again milliseconds after t', even if the current potential P_v is greater than the threshold. This phenomenon is known as the refractory period, in which the neuron is still reluctant to fire. Figure 4 illustrates the $\theta_v\left(t-t'\right)$ function that models the refractory phenomenon.

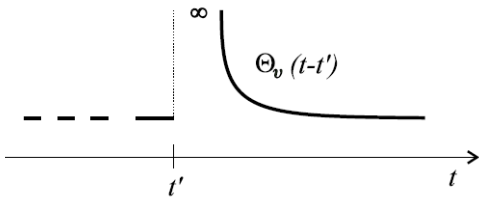

FIGURE 4. Typical form of the threshold function of a biological neuron [13].

A SNN consists of a set of spiking neurons and a set of synapses connecting neurons between layers. The firing potential of a neuron v at time t, of a SNN network, is given by (2), where Γ_u represents the set of neurons in the layer before neuron v, and s is the firing time of neuron u. The variables d_{uv}^k and w_{uv}^k are, respectively, the delay time and weight associated with the synapse k between neuron u and neuron v [10], [13].

$$P_v(t) = \sum_{u\in\Gamma_u}\sum_{k} w_{uv}^k \epsilon\left(t-s-d_{uv}^k\right) \tag{2}$$

For a SNN, a typical response function is the exponential, defined in (3), and which represents the rise and decay of the potential of a postsynaptic neuron (EPSP), as illustrated in Figure 3. In this equation, τ is the decay time constant of the

membrane potential [10].

$$\epsilon(t) = \begin{cases} \frac{t}{\tau} e^{(1-t/\tau)}, & t > 0 \\ 0, & t < 0 \end{cases} \tag{3}$$

Algorithm 1 shows the algorithm for calculating the firing potential, where K is the total synapses, and the parameters t_v, t_u, d_{uv} and w_{uv}, are respectively, the firing time of the neuron v, the firing time of the presynaptic neuron u, the delay time between the neurons u and v, and the synapse weight.

Algorithm 1 Algorithm to Calculate the Firing Potential of the Neuron v.

```
P_v ← 0
while k <= K
    t ← t_v − t_u − d_uv
    if t > 0
        ε ← (t/τ) e^(1−t/τ)
    else
        ε ← 0
    P_v ← P_v + w_uv
```

For this work, the Izhikevich neuron model was used, developed by Eugene M. Izhikevich in 2003, based on the Integrate-and-Fire and Hodgkin-Huxley models. A spiking neural network built from this model can be easily implemented in a hardware device like FPGA, for its energy efficiency [14].

A. IZHIKEVICH NEURON MODEL

The Izhikevich model is computationally simple and allows the simulation of firing patterns of real biological neurons. It is a realistic model like the Hodgkin-Huxley model, but just as efficient as the Integrate-and-Fire model. Basically, the Izhikevich model reproduces the behavior of pulses from known types of cortical neurons, as illustrated in Figure 5 [15].

The Izhikevich model is represented by the ordinary differential equations (4) and (5).

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{4}$$

$$u' = a\,(bv - u) \tag{5}$$

In equations (4) and (5), the variable v represents the membrane potential of the neuron and u represents a membrane recovery variable, responsible for the activation of ionic currents of K^+ and inactivation of ionic currents of Na^+, and that gives negative feedback to v. After the spike reaches its apex, around +30 mV, the membrane voltage and recovery variable are reset, according to (6). The values of the synaptic current, or injected DC current, are provided by the variable I. Figure 6 shows the behavior of the potential, according to the Izhikevich model.

$$\text{if } v \geq 30 \text{ mV} = \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{6}$$

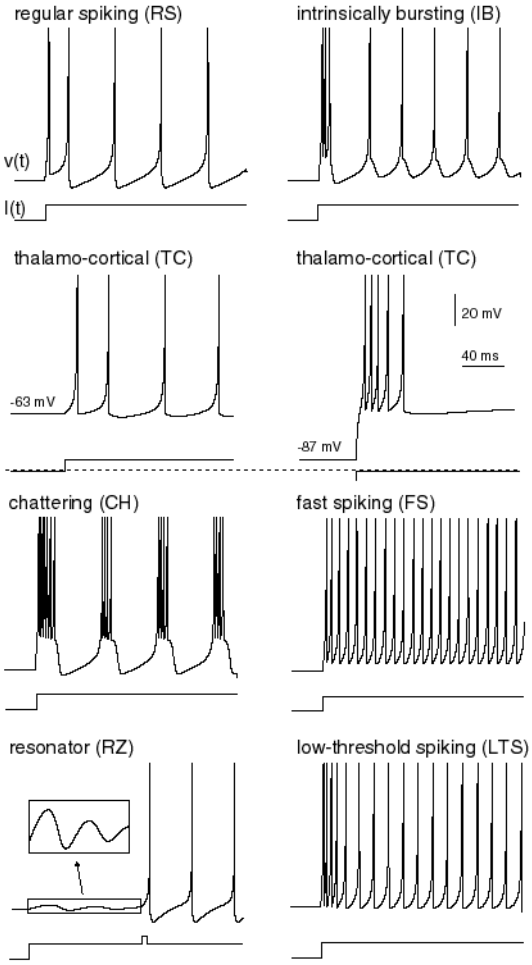

FIGURE 5. Known types of firing responses of cortical neurons, reproduced by the Izhikevich model [15].

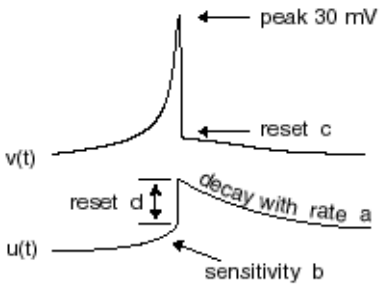

FIGURE 6. Membrane potential according to Izhikevich model [15].

The choice of parameters a, b, c and d, in (4), (5) and (6), result in various intrinsic firing patterns like those of the neocortical neurons in Figure 5. Each parameter modifies a behavior of the Izhikevich neuron:

- The parameter a describes the timescale of the recovery variable u. Smaller values result in slower recovery. A typical value is $a = 0.02$.

- The parameter b describes the sensitivity of the recovery variable u to subthreshold fluctuations in membrane potential v. Higher values couple v and u, resulting in possibly strong subthreshold oscillations and low-threshold peak dynamics. A typical value is $b = 0.2$.
- The parameter c describes the after-spike reset value of the membrane potential v, caused by conductances K^+ of high threshold fast. A typical value is $c = -65$ mV.
- The parameter d describes the after-spike reset of the recovery variable u, caused by conductances Na^+ and K^+ of high threshold slow. A typical value is $d = 2$.

III. DISCRETE COSINE TRANSFORM

The discrete cosine transform (DCT) is used in several applications of signal digital processing, mainly in the compression of audio and video signals. For example, DCT is the basis for multimedia encoding standards such as JPEG, MPEG-2, MPEG-4 and H.264/AVC [16].

DCT is a real orthogonal transform that maps a signal to real coefficients [17]. The coefficients are found through (7) [18] for $0 \leq k \leq N-1$.

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos \frac{\pi \left(n + \frac{1}{2}\right) k}{N} \tag{7}$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & 1 \leq k \leq N-1 \end{cases} \tag{8}$$

The inverse DCT is defined by (9) [18] for $0 \leq n \leq N-1$.

$$Cx(n) = \sum_{k=0}^{N-1} \alpha(k) C(k) \cos \frac{\pi \left(n + \frac{1}{2}\right) k}{N} \tag{9}$$

An important property of DCT defines that most of the signal energy is highly concentrated in a few coefficients, allowing the development of efficient video and audio compression techniques. Figure 7(b) shows the $C(k)$ coefficients of the DCT of a line of a digitized television signal, corresponding to the signal $x(n)$ of Figure 7(a), where it is observed that the energy signal of the samples is distributed approximately uniformly, having its energy strongly concentrated in the first coefficients of the transform, in the frequency domain. The first lowest frequency coefficient is known as the DC coefficient and the others are the AC coefficients. Although each DCT coefficient carries different signal information, it is in the DC coefficient that the average intensity of the transformed signal is concentrated [18], [19].

There are few studies related to the application of DCT in SNN. In general, the application of DCT in spiking neural networks is associated with the transformation of signals into coefficients that are processed by SNN. In the work of Garg, Chowdhury and Roy [20] a scheme for image coding using DCT was proposed, decomposing the spatial information into a set of orthogonal weighted values that are processed by SNN. On the other hand, the work by Wu *et al.* [21] proposed a SNN capable of performing DCT for visual processing, allowing the extraction of image characteristics.

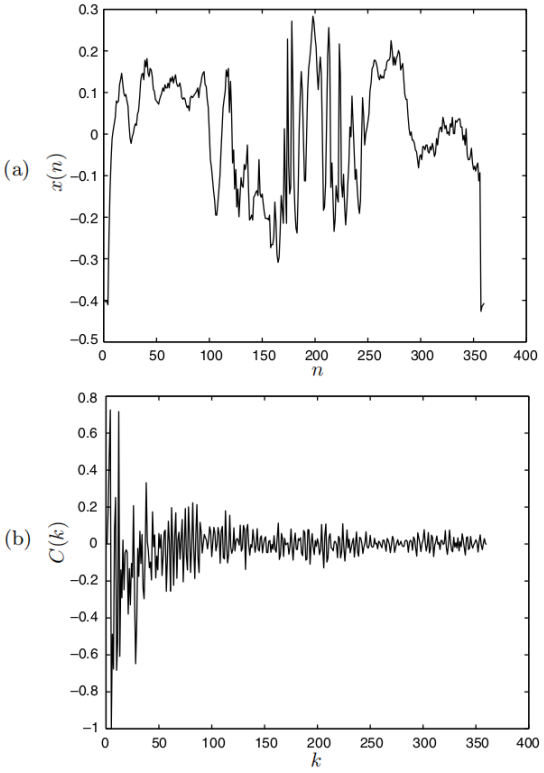

FIGURE 7. DCT of a video signal: (a) discrete-time video signal $x(n)$; (b) the DCT of $x(n)$ [18].

IV. PAIRING

Pairing functions are unique functions and bijections that map a pair of coordinates, represented by values non-negative integers to a single coordinate, which also has a value non-negative integer [22].

$$f : \mathbb{Z}^* \times \mathbb{Z}^* \rightarrow \mathbb{Z}^* \tag{10}$$

Pairing functions are important in the fields of computing and mathematics. They are useful in computational applications such as enumerating complete binary trees and finite-length sequences. They are also applied to optimize data storage, so that two binary numbers of size n can be represented by a single binary number of size less than or equal to $2n$. They are useful in shading applications, mapping systems, and image rendering [23], [24]. The work of Solís-Rosas *et al.* [22], employed a pairing function as a length encoder, allowing to increase the compression rate of medical images such as magnetic resonance, X-rays and computed tomography. The work of Reddaiah [25] showed the importance of the pairing function in cryptographic systems.

A. CANTOR'S PAIRING FUNCTION

One of the well-known functions of pairing is the Cantor's function which maps each pair of positive integers (x, y) through (11), into a single positive integer $p(x, y)$ [24].

$$p(x, y) = \frac{1}{2}\left(x^2 + 2xy + y^2 - x - 3y + 2\right) \tag{11}$$

Considering the pairing function for non-negative integers $c(x, y) = p(x + 1, y + 1) - 1$, (11) can be replaced by (12).

$$c(x, y) = \frac{1}{2}\left(x^2 + 2xy + y^2 + 3x + y\right) \tag{12}$$

The inverse of the pairing function, $(x, y) = f^{-1}(z)$ is given by (13).

$$c^{-1}(z) = \left(z - \frac{w(w+1)}{2}, \frac{w(w+3)}{2} - z\right) \tag{13}$$

where

$$w = \left\lfloor \frac{-1 + \sqrt{1 + 8z}}{2} \right\rfloor \tag{14}$$

Graphically, Cantor's pairing function assigns consecutive numbers to points along the diagonals in the plane, as shown in Figure 8 [26].

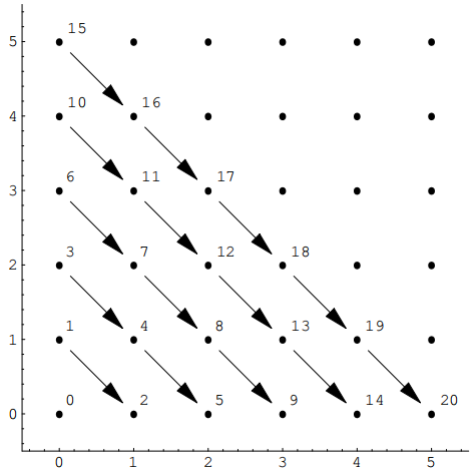

FIGURE 8. Graph of the Cantor's pairing function [26].

Although simple and less complex, the Cantor's pairing function has the disadvantage of losing efficiency with the use of higher dimensional values [25].

B. ELEGANT PAIRING FUNCTION

To compensate for the disadvantage of the Cantor's pairing function, the elegant pairing function is used, also known as the Szudzik's pairing function [26]. Figure 9 graphically shows how the elegant pairing function assigns consecutive numbers to points along the edge of the squares.

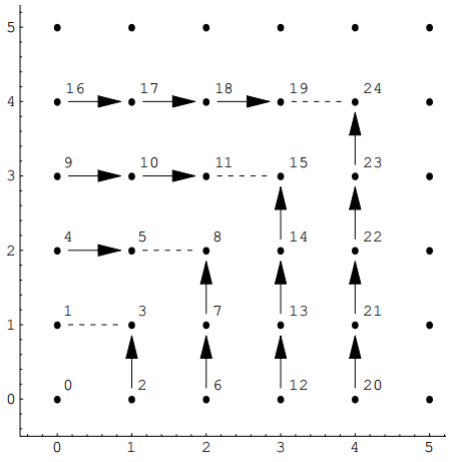

FIGURE 9. Graph of the elegant pairing function (Szudzik) [26].

For integer values x and y, the elegant pairing function generates a non-negative integer $p(x, y)$, through (15) [22], [26].

$$p(x, y) = \begin{cases} y_p^2 + x_p, & x_p < y_p \\ x_p^2 + x_p + y_p, & x_p \geq y_p \end{cases} \tag{15}$$

where

$$x_p = \begin{cases} 2x, & x \geq 0 \\ (-2x) - 1, & x < 0 \end{cases} \tag{16}$$

and

$$y_p = \begin{cases} 2y, & y \geq 0 \\ (-2y) - 1, & y < 0 \end{cases} \tag{17}$$

In the inverse function of elegant pairing, a pair (x, y) is associated with a non-negative integer z, as shown in (18) for $f : Z \to (x, y)$.

$$Z = \begin{cases} \left(z - z_{q^2}, z_q\right), & z - z_{q^2} < z_q \\ \left(z_q, z - z_{q^2} - z_q\right), & z - z_{q^2} \geq z_q \end{cases} \tag{18}$$

where

$$z_q = \lfloor \sqrt{z} \rfloor \quad z_{q^2} = \lfloor \sqrt{z} \rfloor^2 \tag{19}$$

Equations (20) and (21) normalizes the pair (x, y) to allow negative integers.

$$x = \begin{cases} x/2, & x \bmod 2 = 0 \\ (x + 1)/(-2), & x \bmod 2 \neq 0 \end{cases} \tag{20}$$

$$y = \begin{cases} y/2, & y \bmod 2 = 0 \\ (y + 1)/(-2), & y \bmod 2 \neq 0 \end{cases} \tag{21}$$

For this work, was used the elegant pairing function, due to the need to store large positive and negative integers.

V. RESNN-DCT

As explained earlier, the interest in the use of neural networks in embedded applications has grown a lot in recent years, directing research into solutions that overcome the limitations imposed by the hardware itself, such as low energy efficiency and scalability.

This work proposes a methodology, henceforth identified by ReSNN-DCT (abbreviation of Reduction of SNN by DCT), which allows to reduce a spiking neural network (SNN), applying the discrete cosine transform (DCT) in the synapses of its neurons, and the elegant pairing function in the neurons of the hidden layers, improving the scalability of this type of neural network. Figure 10 presents ReSNN-DCT, where blocks 1, 2 and 3 correspond to the stages of the direct ReSNN-DCT, responsible for reducing the structure of the spiking neural network. The stages of the inverse ReSNN-DCT, corresponding to blocks 4, 5 and 6, are part of the pre-processing performed before the calculation of the firing potential of the neuron, that was described in section II. With the coefficients of the restored synapses, it is possible to run the Algorithm 1 (section II), finding the firing potential of each SNN neuron.

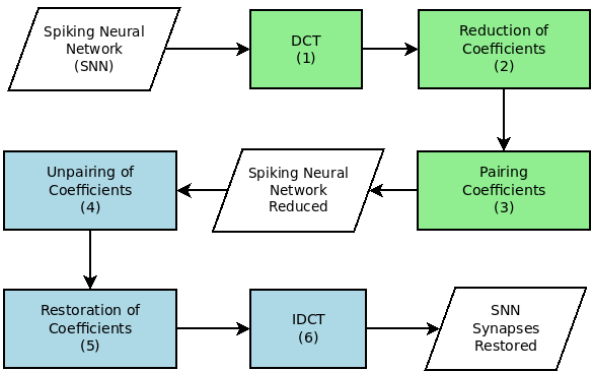

FIGURE 10. ReSNN-DCT diagram for spiking neural networks.

In the direct ReSNN-DCT, shown in Figure 10, DCT is applied to the weights and also on delay times in synapses of the spiking neurons the layer to be reduced. Next, the coefficients with lower energy are eliminated, allowing to reduce the number of DCT matrix coefficients. In the last stage of the reduction process, the neuron synapse coefficients are paired, allowing to reduce the number of neurons in the layer by a factor of 1/2. Before performing the calculation of the firing potential of the neurons in the layer, it is necessary to perform the inverse ReSNN-DCT, which has 3 stages. The first stage, corresponding to unpairing, decodes the paired coefficients, based on the total number of neurons of the original layer. In the second stage of the inverse ReSNN-DCT, the coefficients eliminated in the direct ReSNN-DCT are restored from the total energy eliminated. Finally, with the coefficients restored, IDCT is applied to restore the values of weights and delay times of the synapses that are used in the calculation of the firing potential.

A. APPLYING THE DCT

Based on the Izhikevich model, each neuron synapse contains the weight information, which defines the weighted force for the firing potential, and of the delay time, so that the firing of the presynaptic neuron reaches the neuronal membrane.

Considering the synapse matrix $syn(n)$ in (22), with the values of weights and delay times, where n is the index of the synapse, with $\{n \in \mathbb{N} \mid 0 \leq n \leq N-1\}$, and N is the total synapses.

$$syn(n) = \begin{bmatrix} w(0) & d(0) \\ w(1) & d(1) \\ w(2) & d(2) \\ \cdots & \cdots \\ w(N-1) & d(N-1) \end{bmatrix} \tag{22}$$

Equation (7) from the DCT, presented in section III, is applied to the matrix $syn(n)$, obtaining the matrix

$$C(k,r) = \begin{bmatrix} C_w(0) & C_d(0) \\ C_w(1) & C_d(1) \\ C_w(2) & C_d(2) \\ \cdots & \cdots \\ C_w(N-1) & C_d(N-1) \end{bmatrix} \tag{23}$$

where

$$C_w(k) = \alpha(k) \sum_{n=0}^{N-1} syn(n,1) \cos \frac{\pi \left(n + \frac{1}{2}\right) k}{N} \tag{24}$$

and

$$C_d(k) = \alpha(k) \sum_{n=0}^{N-1} syn(n,2) \cos \frac{\pi \left(n + \frac{1}{2}\right) k}{N} \tag{25}$$

In equations (23), (24) and (25), is defined that $\{k \in \mathbb{N} \mid 0 \leq k \leq N-1\}$, r is the index of the column, indicating whether it is a coefficient of weight ($r = 1$) or delay time ($r = 2$), and N is the total synapses.

B. REDUCTION OF SYNAPSES

The first step for reducing the number of synapses, of each neuron of the SNN layer, is to determine which DCT coefficients will be eliminated. In equation (26), the index of the first coefficient to be eliminated is defined, where fr is the reduction factor, with value between the interval $\{fr \in \mathbb{R} \mid 1 < fr < N\}$.

$$ind = \left\lfloor \frac{N}{fr} \right\rfloor + 1 \tag{26}$$

The coefficients $C(k,r)$ are eliminated if the index k is greater than or equal to ind, resulting in the array of values of the remaining coefficients

$$C_{rem}(k,r) = C(k,r) \quad \forall\, k < ind \tag{27}$$

Considering the AC coefficients eliminated, C_{elim}, according to (28), the information of the total amount of energy is stored for later use in the stage for restoring the coefficients

with less energy, as shown in (29), where M is the number of coefficients eliminated.

$$C_{elim}(k,r) = C(k,r) \quad \forall\, k \geq ind \tag{28}$$

$$E_{elim}(r) = \sum_{k=0}^{M-1} C_{elim}(k,r) \tag{29}$$

C. PAIRING THE COEFFICIENTS

In order to reduce the number of neurons in the layer, the elegant pairing function is used to encode the coefficients of each pair of neurons into a single integer, resulting in a paired neuron, as shown in Figure 11, where it is observed that the block of the pairing function uses a pair of coefficients, corresponding to neurons 1 (N1) and 2 (N2), respectively, resulting in an integer value in neuron 1-2. For example, the pair of coefficients (130.87, 126.65) are paired, resulting in the integer value 685129780. This process is repeated until the k coefficient of the two neurons. Thus, the result of this process is a single neuron that stores the coefficients of neurons 1 and 2, in integer values.

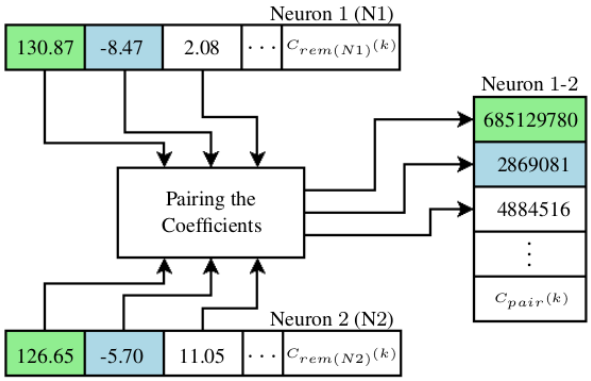

FIGURE 11. Example of pairing the DCT coefficients of the weights of neurons in a layer of SNN.

Considering that the DCT coefficients are real values, a normalization of each coefficient is performed, through the conversion to an integer value, using the ceiling function, as shown in (30). For this work, the constant 100 is multiplied with the coefficient, reducing the precision to two decimal digits.

$$C_{rem}(k,r) = \lceil 100\ C_{rem}(k,r) \rceil \tag{30}$$

As shown in the example of Figure 11, the elegant pairing function is applied to the normalized coefficients of neurons 1 and 2, generating the vector $C_{pair}(k)$ of non-negative integer values.

$$f_{pairing} : \{x, y\} \rightarrow C_{pair}(k,r) \tag{31}$$

where

$$x = C_{rem(N1)}(k,r) \quad y = C_{rem(N2)}(k,r) \tag{32}$$

After the stages of reducing and pairing the coefficients, the SNN layer is reduced by half of its neurons. The number of remaining neurons is defined by (33), where Q is the number of neurons in the SNN layer before applying direct ReSNN-DCT.

$$Q_{red} = \begin{cases} Q/2, & Q \bmod 2 = 0 \\ (Q/2)+1, & Q \bmod 2 \neq 0 \end{cases} \tag{33}$$

D. INVERSE RESNN-DCT

The inverse ReSNN-DCT, which consists of the stages of unpairing and restoring the coefficients, and the application of IDCT, is performed before calculating the firing potential. Algorithm 2 shows the algorithm 1 presented in section II, adapted to the ReSNN-DCT methodology.

Algorithm 2 Algorithm of The Inverse ReSNN-DCT Function.

1 $\{C_w(k), C_d(k)\} \leftarrow f_{\text{inverse pairing}} : C_{pair}(k,r)$
2 $C(v)(k,r) \leftarrow f_{\text{rest coef}} : \{C_w(k), C_d(k), E_{elim}(r), M\}$
3 $\{w_{uv}(n), d_{uv}(n)\} \leftarrow$ `idct`$[C(v)(k,r)]$
4 $P_v \leftarrow 0$
5 `while` n $< N$
6 $\quad t \leftarrow t_v - t_u - d_{uv}(n)$
7 $\quad$ `if` $t > 0$
8 $\quad\quad \epsilon \leftarrow \frac{t}{\tau} e^{(1-t/\tau)}$
9 $\quad$ `else`
10 $\quad\quad \epsilon \leftarrow 0$
11 $\quad P_v \leftarrow P_v + w_{uv}(n)$

In line 1 of the algorithm, the inverse elegant pairing function over the matrix $C_{pair}(k,r)$ is performed, generating the normalized coefficients of the pair of paired neurons. Next, the coefficients are divided by 100 to convert the values into real numbers with two digits of precision, as shown in (34). To calculate the potential, the coefficients of the neuron v are selected.

$$\{C_w(k), C_d(k)\} = \frac{\{C_w(k), C_d(k)\}}{100} \tag{34}$$

In line 2 of the Algorithm 2, the coefficients eliminated during direct ReSNN-DCT are restored. The size of the matrix of eliminated coefficients is $M \times 2$, and the accumulated values, associated with weights and delay times, are represented by $E_{elim}(v)(r)$, where v indicates the neuron and r indicates whether the total energy of the eliminated coefficients is from the weight ($r = 1$) or from the delay time ($r = 2$). The matrix of eliminated coefficients is reconstructed in (35).

$$C_{rest}(k,r) = \begin{bmatrix} c_{w_{rest}}(0) & c_{d_{rest}}(0) \\ c_{w_{rest}}(1) & c_{d_{rest}}(1) \\ c_{w_{rest}}(2) & c_{d_{rest}}(2) \\ \cdots & \cdots \\ c_{w_{rest}}(M-1) & c_{d_{rest}}(M-1) \end{bmatrix} \tag{35}$$

where

$$c_{w_{rest}} = \frac{E_{elim}(v)(1)}{M}, \quad c_{d_{rest}} = \frac{E_{elim}(v)(2)}{M} \tag{36}$$

Thus, the complete matrix of coefficients is given by

$$C(v)(k,r) = \begin{bmatrix} C_w(0) & C_d(0) \\ C_w(1) & C_d(1) \\ \cdots & \cdots \\ C_w(ind-1) & C_d(ind-1) \\ C_{rest}(0,1) & C_{rest}(0,2) \\ C_{rest}(1,1) & C_{rest}(1,2) \\ \cdots & \cdots \\ C_{rest}(M-1,1) & C_{rest}(M-1,2) \end{bmatrix} \quad (37)$$

After restoring the coefficients, IDCT is applied to the matrix $C(v)(k,r)$, shown in (9), as presented in section III. The vector of the weights $w_{uv}(n)$ in (39), is obtained by (38), where $\{n \in \mathbb{N} \mid 0 \leq n \leq N-1\}$, and N is the total number of rows of the matrix $C(v)(k,r)$.

$$w_{uv}(n) = \sum_{k=0}^{N-1} \alpha(k) C(v)(k,1) \cos \frac{\pi \left(n+\frac{1}{2}\right) k}{N} \quad (38)$$

$$w_{uv}(n) = \begin{bmatrix} w_{uv}(0) \\ w_{uv}(1) \\ \cdots \\ w_{uv}(N-1) \end{bmatrix} \quad (39)$$

The vector of delay times $d_{uv}(n)$ in (41), is obtained by (40), where $\{n \in \mathbb{N} \mid 0 \leq n\ leqN-1\}$, and N the total number of rows of the matrix $C(v)(k,r)$.

$$d_{uv}(n) = \sum_{k=0}^{N-1} \alpha(k) C(v)(k,2) \cos \frac{\pi \left(n+\frac{1}{2}\right) k}{N} \quad (40)$$

$$d_{uv}(n) = \begin{bmatrix} d_{uv}(0) \\ d_{uv}(1) \\ \cdots \\ d_{uv}(N-1) \end{bmatrix} \quad (41)$$

VI. EXPERIMENTAL EVALUATION

To evaluate the feasibility of the ReSNN-DCT methodology, experiments were performed which consisted in generating random weights and delay times of synapse samples, and applying the proposed methodology on these samples, comparing the frequencies generated by the Izhikevich model of spiking neuron, before and after ReSNN-DCT.

A. TESTS ENVIRONMENT

The experiments were performed on the Linux platform, on an AMD Rizen 5 Radeon Vega 3.7 GHz computer, with 12 GBytes of RAM and an AMD Radeon Picasso graphics card. For the implementation and simulation of the algorithms of the ReSNN-DCT methodology, was used the Octave software.

B. SIMULATIONS AND RESULTS

Table 1 shows the values of the parameters of the Izhikevich spiking neuron model used in the simulations. These parameters allow simulating the standard response of the RS neuron (Regular Spiking), which is the most typical neuron in the cortex, and whose characteristic allows generating peaks with regular intervals, whose frequency can be controlled by the DC current injected into the synapses [15]. This feature is useful for simulation, as it simplifies the comparison of the SNN network response, before and after the application of the ReSNN-DCT methodology, in terms of frequency.

TABLE 1. Parameters of the pattern of RS spike (Regular spiking) for the Izhikevich model.

Parameter	Variable	Value
Time scale of u	a	0.02
Sensitivity of u	b	0.2
Restart potential	c	-65
Reset rate	d	8
Processing time (ms)	T	500
Rate of change	dT/dt	0.5
Firing time (ms)	t_d	50
Recovery time (ms)	t_r	50
Decay rate	τ	14

For the experimental evaluation, the coding rate in frequency of the neuron was considered, based on (42), where n_p is the number of spikes, and T_{pikes} is the time interval between one spike and another [27].

$$f = \frac{n_p}{T_{pikes}} \quad (42)$$

The simulations generated 18068 data samples, with 8960 samples referring only to tests of the reduction stage of synapses, and 9108 samples referring to the complete stages of the proposed methodology, including the direct and inverse stages of ReSNN-DCT. Spiking neurons were simulated with the number of synapses represented by the set $S = \{syn \in \mathbb{N} \mid 0 \leq syn \leq N-1\}$, or $S = \{0, 1, 2, \ldots, N-1\}$. For 5 synapses ($syn = 5$), simulations of the ReSNN-DCT methodology were performed, corresponding the reductions $1/2$ and $1/4$, that is, the number of synapses was reduced by half and by factor of $1/4$, respectively. For 10 synapses ($syn = 10$), simulations were performed using the ReSNN-DCT methodology, corresponding to reductions of $1/2$, $1/4$, $1/6$ and $1/8$. For more than 10 synapses ($syn > 10$), reductions of $1/2$, $1/4$, $1/6$, $1/8$ and $1/10$ of the number of synapses of the spiking neuron were performed. All simulations were repeated until the number of samples mentioned above was reached.

Table 2 shows the evolution of synapse reductions, considering reduction factors from $1/2$ to $1/10$, where the "Syn" column informs the number of synapses remaining after the application of the reduction stage of the coefficients of the ReSNN-DCT methodology, the column "f_2" represents the resulting frequency, and the column "$f_1 - f_2$" corresponds to the difference, in percentage, of the response in frequency of the neuron, before and after the application of ReSNN-DCT.

In Table 2, it is observed that for the 5 simulations of a neuron with 40 synapses, using the ReSNN-DCT

TABLE 2. Reduction of 40 synapses of an Izhikevich neuron with a response in frequency of 85.5777 Hz (61 firing spikes).

Reduction factor	Syn	f_2 (Hz)	$f_1 - f_2$ (%)	Weight energy (%)	Delay time energy (%)
1/2	20	85.3435	0.27	98.14	90.52
1/4	10	85.5777	0	91.17	89.53
1/6	6	85.5777	0	95.31	92.23
1/8	5	85.5777	0	97.02	92.58
1/10	4	85.5777	0	96.04	96.40

methodology, a difference of 0.27% from the response in frequency of the neuron occurred in only one simulation. It can also be seen that most of the energy of the coefficients, associated with weight and delay time are greater than 90%, with the exception of the second simulation, where the energy of the delay time coefficients was 89.53%. In Table 3 the same behavior is observed, occurring only in two simulations, with a difference between the frequencies, before and after ReSNN-DCT, of 0.08%. Regarding the energies of the stored coefficients, the percentage was above 90%.

TABLE 3. Reduction of 495 synapses of an Izhikevich neuron with a response in frequency of 471.5663 Hz (415 firing spikes).

Reduction factor	Syn	f_2 (Hz)	$f_1 - f_2$ (%)	Weight energy (%)	Delay time energy (%)
1/2	247	471.9551	0.08	96.87	90.71
1/4	123	471.5663	0	91.14	97.11
1/6	82	471.5663	0	92.99	94.77
1/8	61	471.5663	0	92.12	92.77
1/10	49	471.9551	0.08	91.50	94.45

With respect to the frequencies $f1$ and $f2$, respectively, before and after ReSNN-DCT, only 1.84% of the samples, from the total of 18068 simulations, showed any difference between them, and a difference above 1% between the values of the frequencies occurred in only 0.02%.

As shown in section V, the total amount of energy of the eliminated coefficients, during the reduction stage, is used in the restoration stage, through the distribution of this total amount by the number of eliminated coefficients, as shown in (35). This is possible because in general, less than 10% of the energy is concentrated in the eliminated coefficients. Despite this, in 4.44% of the simulations, the concentration of energy reached percentages higher than 20% and lower than 45%, as shown in Table 4. For example, 287 (3.20%) of the 8960 simulations, referring to the reduction stage, concentrated between 20% and 25% of the DCT energy in the eliminated coefficients.

After considering the stage of reducing the number of coefficients, it is important to observe how this impacts the data storage capacity of the network structure. Considering

TABLE 4. Distribution range of the energy percentage of the eliminated coefficients above 10%.

Energy >	Number of Simulations	%
20%	287	3.20
25%	86	0.96
30%	17	0.19
35%	5	0.06
40%	3	0.03

as example, the information in Table 3, for a neuron with 495 synapses, without reduction, a 495 $\times$ 2 array would be needed to store 990 values of the real type. With the reduction by a factor of 10, the storage is reduced to 98 values of real coefficients in a 49×2 array. Table 5 presents a comparison of the storage capacity of the coefficients, for some simulation samples, before and after the application of the reduction stage of the ReSNN-DCT methodology, where the column "Syn" corresponds to number of synapses.

TABLE 5. Comparison of the storage capacity of the SNN network with the ReSNN-DCT methodology, after the reduction stage, for the reduction factors of 1/2 (worst case) and 1/10 (best case).

Syn	Before ReSNN-DCT	After reduction	
		Factor 1/2	Factor 1/10
20	40	20	4
55	110	54	10
150	300	150	30
515	1030	514	102
785	1570	784	156

It is observed in Table 5, that for a neuron with 785 synapses, knowing that each neuron has two real values (weight and delay time), 1570 coefficients are stored, without the application of ReSNN-DCT. Applying the reduction factor of 1/2 for worst case, the number of stored coefficients was reduced to 784. Considering the best case of reduction by a factor of 1/10, it was possible to store only the 156 coefficients that possess more than 90% of the DCT energy. It is important to note that for the values in the last two columns, 2 units must be added, which correspond to the stored values of the total amount of energy and of the amount of coefficients eliminated. Thus, the values of the last rows of the table, for reduction factors of 1/2 and 1/10, respectively, are 786 and 158, after ReSNN-DCT.

Elegant pairing allows a decrease in 2$\times$ the amount of information to be stored, hence reducing the amount of SNN neurons. As shown in Table 6, the number of coefficients is substantially reduced, considering the number of synapses per neurons. In this table, the columns "NN" and "Syn" correspond, respectively, to the number of neurons per layer and the number of synapses per neuron. As an example, in row 2 of the table, for a layer with 50 neurons, knowing that each neuron has 50 synapses, an array of 2500 $\times$ 2 will

TABLE 6. Comparison of the storage capacity of the SNN network with the ReSNN-DCT methodology, after the stage of elegant pairing, for the reduction factors of 1/2 (worst case) and 1/10 (best case).

NN	Syn	Before ReSNN-DCT	After pairing	
			Factor 1/2	Factor 1/10
20	40	1600	400	80
50	50	5000	1250	250
75	50	7500	1875	370
90	20	3600	900	180
100	35	7000	1700	300

TABLE 7. Processing times after application of the ReSNN-DCT methodology for a neuron with 40 synapses.

NN	Processing times (s)			
	Before ReSNN-DCT	After ReSNN-DCT	Time increase	
			(ms)	%
20	0.5906	0.5917	1.1361	0.19
35	1.0152	1.0170	1.7972	0.18
40	1.1224	1.1564	34.0221	3.03
45	1.2523	1.3124	60.0982	4.80
60	1.6899	1.7469	56.9829	3.37
67	1.9028	2.5845	681.7221	35.83
80	2.2093	2.2873	77.9841	3.53
91	2.5965	3.0737	477.2130	18.38
100	2.7630	2.8414	78.4021	2.84

be needed to store all the 5000 real coefficients of the weights and delay times of the layer. After executing the elegant pairing function ReSNN-DCT, considering the reduction per worst case (1/2), the matrix would be reduced to the size of 200×2, capable of storing 1250 integer values, corresponding to the coefficients of weights and delay times. For the best-case reduction, by a factor of 1/10, the array is reduced to the size of 40×2, capable of storing 250 integer-type values. For the reduction factor of 1/2, the gain of storage space, in terms of the amount of values, is 75%, and for the factor of 1/10, the gain is 95%.

Another important point to be analyzed is the processing time of the SNN network, after the reduction of its layers by the ReSNN-DCT methodology. Naturally, there will be an increase in the execution time of the potential calculation algorithm, shown in Algorithm 2, in section V, where the stages of unpairing and restoration of the coefficients and the stage of IDCT on the restored coefficients are executed. Table 7 shows the processing times for some of the simulations performed, where the time values are in seconds, but in the fifth column the increment time values are informed in milliseconds. In the sixth column, the percentage of the increase in the processing time after ReSNN-DCT is informed. As observed in the table, only the simulations with 67 and 100 neurons had a time increase greater than 5%, reflecting what happened in the simulations performed during the experimental evaluation.

TABLE 8. Range of increase in processing time after application of the ReSNN-DCT methodology.

Range of increase in processing time (a_t)	%
$a_t < 5\%$	74.76
$5\% < a_t < 10\%$	19.89
$10\% < a_t < 20\%$	4.83
$20\% < a_t < 40\%$	0.49
$a_t > 50\%$	0.03

In order to analyze the processing time, in addition to the number of simulations mentioned, referring to the direct and inverse stages of ReSNN-DCT, another 9108 simulations were performed, totaling 18216. In general, in the inverse ReSNN-DCT, there is an increase of less than 5% on the processing time for more than 74% of the simulations performed, as shown in Table 8. At 19.89% of the simulations, the increment is greater than 5% and less than 10%.

VII. CONCLUSION

This work presented a methodology proposal, based on the discrete cosine transform (DCT) and on the elegant pairing technique, for the reduction of layers of a spiking neural network, which uses the Izhikevich neuron model.

One of the objectives of this work was to demonstrate that the amount of synapses per neuron can be reduced, concentrating most of the energy of the weights and delay times in the remaining coefficients, generated by DCT during the direct ReSNN-DCT. One of the interesting features of DCT is that, as few coefficients concentrate most of the signal energy, it was possible to equally distribute the total AC energy among the eliminated coefficients. In the simulations, it was observed that for the worst case of reduction of synapses (1/2), using ReSNN-DCT, the amount remaining was 50%, and in the best case of reduction (1/10), the amount of remaining synapses was 10%. But in both situations, the energy of the DCT coefficients was above 90%, meaning the possibility of complete restoration of synapse weights and delays. Thus, it is necessary to store only the remaining coefficients and the information about the total energy and the amount of the eliminated coefficients, for use in the pre-processing of the calculation of the firing potential, in the inverse ReSNN-DCT.

It has also been shown that it is possible to reduce the number of neurons in the SNN network layers by applying elegant pairing. This technique allowed to encode coefficients of two neurons of the same layer, pairing in a single integer-type value. In order to be able to use the pairing technique, it was necessary to reduce the precision of the DCT coefficients by two decimal digits. However, this reduction in the precision of the real coefficients does not affect the restoration of the real values of the coefficients, allowing later to perform IDCT on these coefficients, restoring the weights and delay times, in the inverse ReSNN-DCT. In the simulations, a reduction of 75% for the worst case (1/2) and 95% for the best

case (1/10) was achieved, after DCT and elegant pairing. This demonstrates that it is possible to increase the capacity of the neural network by at least 75%, considering the storage capacity of the coefficients, which correspond to the synapse weights and delays.

In conclusion, the ReSNN-DCT methodology proved to be effective in reducing the number of synapses and neurons, allowing for scaling the memory storage capacity, reducing the amount of coefficients needed to store. Despite the increase in processing time for the reduced network, the ReSNN-DCT methodology proved to be efficient in keeping this increase in time below 5%, in more than 74% of the simulated cases.

REFERENCES

[1] S. Paul and L. Singh, "A review on advances in deep learning," in *Proc. IEEE Workshop Comput. Intell., Theories, Appl. Future Directions (WCI)*, Dec. 2015, pp. 1–6.

[2] J. Zhang, C. Tao, and P. Wang, "A review of soft computing based on deep learning," in *Proc. Int. Conf. Ind. Inform.-Comput. Technol., Intell. Technol., Ind. Inf. Integr. (ICIICII)*, Dec. 2016, pp. 136–144.

[3] Q. Yu, C. Wang, X. Ma, X. Li, and X. Zhou, "A deep learning prediction process accelerator based FPGA," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 1159–1162.

[4] G. Lacey, G. W. Taylor, and S. Areibi, "Deep learning on FPGAs: Past, present, and future," 2016, *arXiv:1602.04283*.

[5] S. Yin, S. K. Venkataramanaiah, G. K. Chen, R. Krishnamurthy, Y. Cao, C. Chakrabarti, and J. Sun Seo, "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2017, pp. 1–5.

[6] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "DLAU: A scalable deep learning accelerator unit on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 3, pp. 513–517, Mar. 2017.

[7] Y. Wang, L. Xia, T. Tang, B. Li, S. Yao, M. Cheng, and H. Yang, "Low power convolutional neural networks on a chip," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 129–132.

[8] I. Grout, *Digital Systems Design With FPGAs and CPLDs*, 1st ed. Oxford, U.K.: Elsevier, 2008.

[9] E.-G. Mallorquí, *Digital System for Spiking Neural Network Emulation*. Barcelona, Espanha: Universitat Politècnica de Catalunya, 2017.

[10] N. Singh, *Training Algorithms for Networks of Spiking Neurons*. Austin, TX, USA: Texas A&M Univ., 2014.

[11] L. Lapicque, "Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarisation," *J. Physiol. Pathol. Gen.* vol. 9, pp. 620–635, Jan. 1907.

[12] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952, doi: 10.1113/jphysiol.1952.sp004764.

[13] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608097000117

[14] B. Han, A. Sengupta, and K. Roy, "On the energy benefits of spiking deep neural networks: A case study," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 971–976.

[15] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[16] S. Li, A. Karrenbauer, D. Saupe, and C.-C. J. Kuo, "Recovering missing coefficients in dct-transformed images," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 1537–1540, doi: 10.1109/ICIP.2011.6115738.

[17] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[18] P. Diniz, S. Netto, and E. D. Silva, *Digital Signal Processing: System Analysis and Design*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[19] S. Ong, S. Li, K. Wong, and K. Tan, "Fast recovery of unknown coefficients in DCT-transformed images," *Signal Process., Image Commun.*, vol. 58, pp. 1–13, Oct. 2017.

[20] I. Garg, S. S. Chowdhury, and K. Roy, "DCT-SNN: Using DCT to distribute spatial information over time for learning low-latency spiking neural networks," 2020, *arXiv:2010.01795*.

[21] Q. Wu, T. Mcginnity, L. Maguire, A. Ghani, and J. Condell, "Spiking neural network performs discrete cosine transform for visual images," in *Proc. Int. Conf. Intell. Comput.*, Sep. 2009, pp. 21–29.

[22] A. Solis-Rosas, S. Canchola-Magdaleno, and M. García-Ramírez, "An enhanced run length encoding using an elegant pairing function for medical image compression," *Int. J. Comput. Sci. Softw. Eng.*, vol. 8, pp. 2409–4285, May 2019.

[23] K. W. Regan, "Minimum-complexity pairing functions," *J. Comput. Syst. Sci.*, vol. 45, no. 3, pp. 285–295, Dec. 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002200009290027G

[24] M. P. Szudzik, "The Rosenberg-strong pairing function," Tech. Rep., 2019.

[25] B. Reddaiah, "A study on pairing functions for cryptography," *Int. J. Comput. Appl.*, vol. 149, no. 10, pp. 4–7, Sep. 2016.

[26] M. Szudzik, "An elegant pairing function," in *Proc. Wolfram Sci. Conf.*, 2006, vol. 45, no. 3, pp. 1–12. [Online]. Available: http://szudzik.com/ElegantPairing.pdf

[27] K. Ahmed, "Brain-inspired spiking neural networks," in *Biomimetics*, M. K. Habib and C. Martín-Gómez, Eds. Rijeka, Croatia: InTechopen, 2020, ch. 5, doi: 10.5772/intechopen.93435.

FRANCISCO DE ASSIS PEREIRA JANUÁRIO received the Graduate degree in telecommunications engineering from the Institute of Higher Education FUCAPI, Amazonas, Brazil, in 2012, and the master's degree in electrical engineering from the Federal University of Amazonas (UFAM, Brazil), in 2015, where he is currently pursuing the Ph.D. degree in computer science. He is also an Assistant Professor with the Department of Electronics and Computation, Faculty of Technology (FT), UFAM. He is also a Researcher in the areas of embedded systems, artificial intelligence, and robotics.

JOSÉ REGINALDO HUGH CARVALHO received the Graduate degree in electrical engineering from the Federal University of Bahia, Brazil, in 1990, and the master's and Ph.D. degrees in electrical engineering from the State University of Campinas, São Paulo, Brazil, in 1993 and 1997, respectively. From 2001 to 2009, he was in the industry, acting as the P&D Manager in several companies of the Industrial Hub of Manaus, Siemens, and the Genius Institute of Technology, in the area of processing/compression of video and embedded systems. He is currently an Adjunct Professor with the Computing Institute, Federal University of Amazonas (UFAM, Brazil). Currently, he is developing projects in the areas of critical embedded systems, robotics and computational vision, and distributed software development processes.