Improving Cellular Nonlinear Network Computational Capabilities

Víctor M. Preciado

Departamento de Electrónica e Ingeniería Electromecánica,
Universidad de Extremadura,E-06071 Badajoz, Spain
{vpdiaz@unex.es}

Abstract. The Cellular Neural Network (CNN) is a bidimensional array of analog dynamic processors whose cells interact directly within a finite local neighborhood [2]. The CNN provides an useful computation paradigm when the problem can be reformulated as a well-defined task where the signal values are placed on a regular 2-D grid (i.e., image processing) and direct interaction between signal values are limited within a *local neighborhood*. Besides, local CNN connectivity allows its implementation as VLSI chips which can perform image processing based in local operations at a very high speed [5]. In this paper, we present a general methodology to extend actual CNN operations to a large family of useful image processing operators in order to cover a very broad class of problems.

1 Introduction

The Cellular Neural Network Universal Machine (CNN-UM) is a programmable neuroprocessor having real-time power implemented in a single VLSI chip [8]. This neurocomputer is a massive aggregate of regularly spaced nonlinear analog cells which communicate with each other only through their nearest neighbors. Local connectivity allows its implementation as VLSI chips that can operate at a very high speed and complexity [5]. This fact makes the CNN an useful computation paradigm when the problem can be reformulated as a task where the signal values are placed on a regular 2-D grid, and the direct interaction between signal values are limited within a finite local neighborhood [2]. This cellular structure and the local interconnection topology not only resemble the anatomy of the retina, indeed, they are very close to the operation of the eye [10], especially when photosensors are placed over each tiny analog processor. Several of these tiny processors have been placed on a chip, which is also called visual microprocessor or cellular processor. In this paper, a methodology to extend local CNN computation capabilities to a very broad class of global operators is presented. In Section II, the cellular neural network methematical model and the architecture of the CNN-Universal Machine prototyping system is introduced, as well as some standard CNN operators (also called templates). Sections III and IV describe a general method for implementing any type of piecewise-linear output

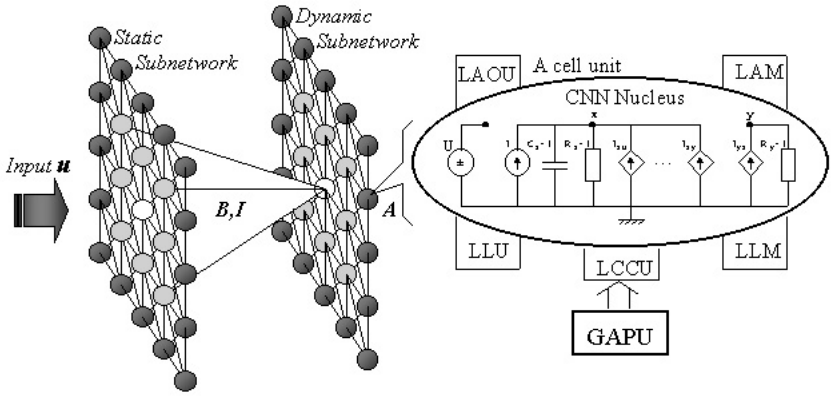

Fig. 1. Block diagram of the CNN.

function and how we can perform integral and wavelet transforms on the CNN-UM framework. In Section V, several examples demonstrating the operation and run-time of the algorithm are discussed. Finally, Section VI gives a brief conclusion.

2 Description of the CNN-UM

The CNN-Universal Machine (CNN-UM) architecture [8] is an *analogic* spatio-temporal array computer wherein analog spatio-temporal dynamics and logic operations are combined in a programmable framework. A computational infrastructure exists in order to interface this computing analog technology to digital systems.

The elementary instructions and the algorithm techniques are absolutely different from any digital computers because elementary instructions (templates) perform complex spatio-temporal nonlinear dynamics phenomena by varying the local interconnection patterns in the array.

2.1 Nonlinear Dynamics of the Network

CNN dynamics can be electronically emulated by a single capacitor which is coupled to neighbouring cells through nonlinear controlled sources as can be seen in Fig. 1. The dynamics of the array can be described by the following set of nonlinear differential equations

$$\frac{d}{dt}x_{i,j}(t) = -x_{i,j}(t) + \sum_{k,l \in N_r} A_{k,l} y_{i+k,j+k}(t) + \sum_{k,l \in N_r} B_{k,l} u_{i+k,j+k}(t) + I \quad (1)$$

$$y(x) = \frac{1}{2}\left[|x-1| - |x+1|\right] \quad (2)$$

Input, state and output, represented by $u_{i,j}, x_{i,j}$ and $y_{i,j}$are defined on $0 \leq i \leq N_1$ and $0 \leq i \leq N_2$ and N_r represents the neighborhood of the cell with a radius r as $N_r = \{(k,j) : \max\{|k-i|, |l-j|\} \leq r\}$. Due to implementability concerns, the template neighborhood radius is generally restricted to be as small as possible, and templates are applied in a space-invariant manner (A and B are the same for every cell).

2.2 Universal Machine Capabilities

The elementary image processing tasks performed on the input data by a single template can be combined to obtain more sophisticated operation mode on the CNN Universal Machine. The machine uses the simple CNN dynamics (1) in a time multiplexed fashion by controlling the template weights and the source of the data inputs for each operation.

The CNN-UM extends the CNN core of each cell in two main parts: *(a)the array of extended CNN cells* and *(b)the Global Analogic Programming Unit (GAPU)*. Several extra elements extend the CNN nucleus (core cell) computational capabilities to a Universal Machine: *(i)4 Local Analog Memories (LAM)*; a few continuous (analog) values are stored in the LAM in each cell, *(ii)4 Local Logic Memories (LLM)*; several binary (logic) values are stored in the LLM in each cell, *(iii)a Local Logic Unit (LLU)*; a simple programmable multi-input/single-output analog operations is executed, the input and output is stored in the LAM, and a *(iv)Local Comunication and Control Unit (LCCU)* which receives the messages from the central (global) "commander", the GAPU, and program the extended cells accordingly.

2.3 Local Processing Capabilities

In image processing applications, two independent input signal arrays $S_1(ij) = u_{i,j}(t)$ and $S_2(ij) = x_{i,j}(0)$ can be mapped onto the CNN, while the output signal array $S_o(ij)$ is associated with $y_{i,j}$. The generic input can be time varying, but the initial state $x_{i,j}(0)$ is usually fixed.

A lot of computational examples about capabilities of image processing on the cellular neural network can be found in the literature [1], [9], [7]. In the following one, it will be illustrated how CNN output are exclusively dependenp on local properties (e.g., average gray level, intensity or texture). The processing is based on using several templates defined as $TEM_k = \left\{A^k; B^k; I^k\right\}$ where $A^k B^k$ are 3×3 convolution matrices and I^k is a biasing scalar.

From an input image Fig. 2(a), we can perform a linear low-pass filtering by convoluting with $TEM_1 = \left\{A^1_{ij} = 0; B^1_{ij} = 0.1(1 + \delta_{2j}\delta_{2i}); I^k = 0, \forall i,j\right\}$. Then we can use the feedback convolution matrix A for thresholding the image by means of $TEM_2 = \left\{A^2_{ij} = 2\delta_{2j}\delta_{2i}; B^1_{ij} = 0; I^k = z^* \ ,\forall i,j\right\}$ where z^* is the value of the threshold. In the following stage, a border extraction template $TEM_2 = \left\{A^2_{ij} = 2\delta_{2j}\delta_{2i}; B^1_{ij} = -1(1 + 9\delta_{2j}\delta_{2i}); I^k = -0.5 \ ,\forall i,j\right\}$ is used. It can be noted how the template extract the binary borders: matrix B controls the dynamics of the CNN convoluting the input image by a linear high-pass filter. The design

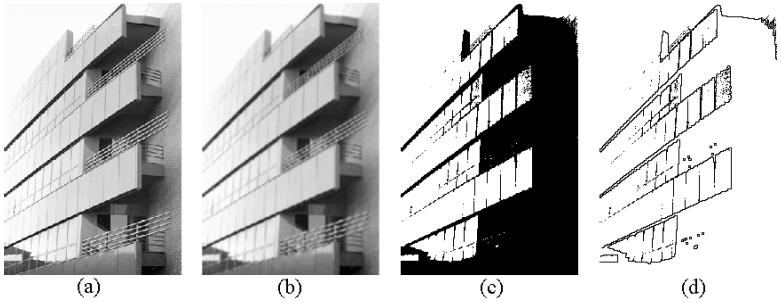

Fig. 2. An example of image processing capabilities on the Cellular Neural Network. (a)-(b) Input image and output after the low pass filter. (c) Thresholding of (b) with template in the normalized scale $[-1, 1]$. (c) Border detection.

of these templates is based in the geometric aspects of the matrices involved in the nonlinear differential equation (1).

In these templates it can be seen how both linear classical filtering and dynamically evolving templates are based on convolution of a image with a 3×3 matrix, this fact makes CNN-UM ill-conditioned for achieving global operations over the whole image (e.g., integral/wavelet transforms or statistical operations).

3 PWL Approximation on the CNN-UM

When considering the VLSI CNN chip model, we deal with the output function (1). In this Section, we present a software-based method to implement any nonlinear output function on current CNNUM chips by superposition of simple piecewise-linear (PWL) saturation blocks in order to approximate any given continuous functions. This approximation is the basis of a novel method for accomplishing global operations on the implemented chip.

3.1 Notation

The following notation is used. δ_{ij} denotes Kronecker delta, $B_{zo,r}$ denotes de open ball $B_{z_o,r} := \{z \in Z : \|z - z_o\| < r\}$, $\|\cdot\|$ is the weighted Euclidean norm defined as $\|z\| = \left(\sum_{i=1}^{n} \omega_i z_i^2\right)^{1/2}, \omega_i > 0$, $\|\cdot\|_\infty$ the weighted infinity norm, $\Delta h_i := h_{i+1} - h_i$, $\Sigma h_i := h_{i+1} + h_i$ and the simbol ' denotes differentiate on variable x.

3.2 Infinite Norm Criterion

In this paper, a superposition $\overline{f}(x)$ of PWL saturation functions is considered

$$\overline{f}(x) = \sum_{i=1}^{\sigma} \left(\frac{1}{b_i} y(a_i x - c_i) + m_i \right) \tag{3}$$

where $y(x)$ is the function defined in (2); $a_i, b_i, c_i, m_i \in \mathbb{R}$ are the parameters of the structure. Basically, (3) it is a nonlinear combination of linear affine lines, $\pi_i := \frac{1}{2}\left[|(a_i x - c_i) - 1| - |(a_i x - c_i) + 1|\right], i \in [1, \sigma]$.

Now, the problem under study can be stated as follows: Given a smooth function $f : S \to \mathbb{R}$, where $S \subset \mathbb{R}$ is compact, we want to design a PWL function $\overline{f}$ that minimizes the error between them in some sense. Formally, given a fixed number ε we want to find the optimal parameter vector $\theta^* = [a_i^*, b_i^*, c_i^*, m_i^*]$ that makes the objective functional $\Im := \left\| f(x) - \overline{f}(x) \right\|_\infty = \varepsilon \quad \forall x \in S$. The method used to obtain the PWL function is based on the following

Theorem 1. *(Minimax) Let $f(x)$ be a function defined in the open subset (x_i, x_{i+1}), $x_i, x_{i+1} \in \mathbb{R}$ and $P_n(x)$ a polynomial with grade n. Then $P_n(x)$ minimizes $\|f(x) - P_n(x)\|_\infty$ if and only if $f(x) - P_n(x)$ takes the value $\varepsilon := \max(|f(x) - P_n(x)|)$ at least in $n+2$ points in the interval (x_i, x_{i+1}) with alternating sign.*

Remark 1. In manifold cases, the condition required by the previous theorem can be analytically expressed. This is the case when $f(x)$ is a concave or convex function in the approximating interval.

Theorem 2. *Let $f(x)$ be a function with $f'' > 0$ in the interval (x_1, x_2), $x_1, x_2 \in \mathbb{R}$ and $P_1(x) := Mx + B$. Then $P_1(x)$[1] minimizes $\|f(x) - P_1(x)\|_\infty$ if and only if $M = \Delta f / \Delta x_1$; $B = \frac{1}{2}\left[f(x_2) + f(x_a) - \Delta f_i / \Delta x_i \, (x_a + x_2)\right]$ where x_a is obtained by solving $f'(x_a) = \Delta f_i / \Delta x_i$*

Proof. It follows from minimax theorem that it must be three points x_1, x_2, x_3 in (x_i, x_{i+1}) which maximize $E(x) := f(x) - P_1(x)$. This condition implies that x_2 is an intermediate point in the interval (x_i, x_{i+1}) with $E'(x)|_{x=x_2} = 0$; this is the same that $f'(x)|_{x=x_2} = M$. Since $f''(x) \geq 0$, $f'(x)$ is a growing function strictly and can equate M only once, this means that x_2 is the only one intermediate point in the interval. Thus $x_1 = x_{i-1}$ and $x_2 = x_i$. Applying the minimax condition, we obtain $f(x_i) - P_1(x_i) = -\left[f(x_2) - P_1(x_2)\right] = f(x_{i+1}) - P_1(x_{i+1})$. By solving these equations we can conclude

$$M = \frac{\Delta f_i}{\Delta x_i}; \; B = \frac{1}{2}\left[f(x_{i+1}) + f(x_a) - \frac{\Delta f_i}{\Delta x_i}\,(x_a + x_{i+1})\right]$$

Corollary 1. *Under the previous conditions the infinity norm is given by the following expression $\varepsilon = f(x) - \left[\frac{\Delta f_i}{\Delta x_i} x + \frac{1}{2}\left(f(x_{i+1}) + f(x_{ai}) - \frac{\Delta f_i}{\Delta x_i}(x_{ai} - x_{i+1})\right)\right]$*

Theorem 3. *Let $f(x)$ be a function with $f''(x) \geq 0$ in the interval (x_a, x_b), $x_a, x_b \in \mathbb{R}$, ε an arbitrary small real number and $\overline{f}(x) = \sum_{i=1}^{\sigma} \pi_i$,where $\pi_i := \frac{1}{2}\left[|(a_i x - c_i) - 1| - |(a_i x - c_i) + 1|\right], i \in [1, \sigma]$; $a_i, b_i, c_i, m_i \in \mathbb{R}, i \in [1, \sigma]$. Then $\overline{f}(x)$ makes $\left\| f(x) - \overline{f}(x) \right\|_\infty = \varepsilon^*$ minimizing σ if the parameters of $\overline{f}(x)$ fulfill the following conditions*

$$\begin{aligned} &a_i = 2/\ \Delta x_i,\ b_i = 2/\ \Delta f_i,\ c_i = \Sigma x_i/\ 2,\ i \in [1, \sigma]; \\ &m_1 = \Sigma f_i/\ 2 - \varepsilon^*,\ m_j = \Sigma f_j/\ 2 - f(x_j) - \varepsilon^*,\ j \in [2, \sigma] \end{aligned} \tag{4}$$

[1] This straight line is called *Chebyshev line* in the literature.

where x_i is defined by the following set of discrete equations

$$\varepsilon^* - \frac{1}{2}\left[x_i + \frac{\Delta f_i}{\Delta x_i}(x_{ai} - x_i) - f(x_{ai})\right] = 0,\ f'(x_{ai}) = \frac{\Delta f_i}{\Delta x_i}$$

Proof. In order to demonstrates this theorem we can express π_i as

$$\pi_i := \begin{cases} m_i - b_i^{-1},\ \forall x \in \left[c_i + a_i^{-1}, \infty\right) \\ m_i + \frac{\Delta f_i}{\Delta x_i}(x - c_i),\ \forall x \in B_{c_i, a_i^{-1}} \\ m_i + b_i^{-1},\ \forall x \in \left(-\infty, c_i - a_i^{-1}\right] \end{cases}$$

Replacing the values of the parameters given in the statement of the theorem

$$\pi_i := \begin{cases} \delta_{1i}\left(f(x_i) - \varepsilon^*\right),\ \forall x \in \left[c_i + a_i^{-1}, \infty\right) \\ \delta_{1i}\left(f(x_i) - \varepsilon^*\right) + \frac{\Delta f_i}{\Delta x_i}(x - x_i),\ \forall x \in B_{c_i, a_i^{-1}} \\ \delta_{1i}\left(f(x_i) - \varepsilon^*\right) + \Delta f_i,\ \forall x \in \left(-\infty, c_i - a_i^{-1}\right] \end{cases}$$

If we consider $x_a \in (x_j, x_{j+1})$ and expand $\overline{f}(x_a)$ taking into account the value of ε^* and the shape of π_i with the parameters given in the theorem it is obtained

$$\begin{aligned} \overline{f}(x_a) &:= \pi_1 + \textstyle\sum_{i=2}^{j-1} \pi_i + \pi_j + \sum_{i=j+1}^{\sigma} \pi_i \\ &= (f(x_1) - \varepsilon^*) + \textstyle\sum_{i=2}^{j-1} \Delta f_i + \left[\frac{\Delta f_j}{\Delta x_j}(x - x_j)\right] \\ &= f(x_j) - \varepsilon^* + \frac{\Delta f_j}{\Delta x_j}(x - x_j) \\ &= \frac{\Delta f_i}{\Delta x_i} x + \frac{1}{2}\left[f(x_{i+1}) + f(x_a) - \frac{\Delta f_i}{\Delta x_i}\left(x_a + x_{i+1}\right)\right] \end{aligned}$$

this is the equation of the *Chebyshev line* that approximated $f(x)$ in the interval (x_j, x_{j+1}) with $\|f(x) - P_1(x)\|_\infty = \varepsilon^*$ as it was expressed in corollary 4.

Corollary 2. *Since the PWL function is continuous in the intervals (x_i, x_{i+1}) and the term $\sum_{i=j+1}^{\sigma} \pi_i$ is null in the expansion of $\overline{f}(x_a)$ performed in the previous proof, it can be affirmed that $\lim_{\Delta x \to 0} f(x_i + \Delta x) = \lim_{\Delta x \to 0} f(x_i - \Delta x)$, and $\overline{f}(x)$ is a PWL continuous function.*

Remark 2. Theorem 3 gives us the possibility of approximating any contiuous function $f(x)$ with $f''(x) \geq 0$ by means of a PWL function as defined by $\overline{f}(x)$ with an arbitrary infinite norm ε^*. Besides, the intervals of the approximation function can be obtained in a forward way if we know the analytical expression of $f(x)$, by means of solving the uncoupled discrete equations stated at the final of theorem 3.

The functional proposed in this paper is an alternative to the $(f(x) - \overline{f}(x))^2$ functional studied in several papers [3], [4]. This cuadratic criterion yields a nonlinear optimization problem characterized by the existence of several local minima. One practical technique used to solve this problem consist in the use of iterative algorithms which produce new random search direction when a local minimum in reached. This fact emphasize the advantage of the direct method based on theorem 3 to design the intervals of approximation opposite to the annealing iterative method needed in the minimization of the cuadratic norm.

3.3 Approximation of Useful Functions

In this point it will be analytically derived the parameter vector θ^* for several concrete function that will be used in the process of performing integral transformation on the CNN. In this approximation it will be used a value $\varepsilon^* = 2^{-7}$ because of the physical implementation of the CNN-UM chip allows an analog accuracy of this magnitude.

In the case of $f(x)$ being a logarithmic or exponential functions the discrete equation in theorem 3 yields the following implicit discrete equations

$$\ln\left(\frac{\Delta x_i}{\Delta \ln_i}\right) + \left(\frac{\Delta \ln_i}{\Delta x_i}\right) x_i - \ln(x_i) - 1 = 2\varepsilon^* \tag{5}$$

$$\frac{\Delta \exp_i}{\Delta x_i}\left[\ln\left(\frac{\Delta \exp_i}{\Delta x_i}\right) + x_i + 1\right] - \exp(x_i) \tag{6}$$

where $\Delta \ln_i = \ln(x_{i+1}) - \ln(x_i)$, $\Delta \exp_i = \exp(x_{i+1})\exp(x_i)$ and $\varepsilon^* = 2^{-7}$. Both equation can be easily solved by standard numerical methods in order to obtain in a sequential and forward way the neighboring points of the intervals that construct the PWL approximating function $\overline{f}(x)$. Similar equation can be obtained for the functions x^2 and $\tan^{-1}(x)$. The parameter vector defined as $\theta^* := [a_1^*, b_1^*, c_1^*, m_1^*, a_2^*, b_2^*, ...]$ can be easily obtained as it was shown in theorem 3

- $ln(x)$; $x_i = \{0.368, 0.608, 1.004, 1.657, 2.735\}$ in the interval limited by the conditions[2] $f(x_1) = -1$, $f(x_\sigma) = 1$.
- $exp(x)$; $x_i = \{-2, -1.5, -1, -0.310, 0.202, 0.610, 0.949, 1.239, 1.699, 1.856, 2\}$
- x^2; $x_i = \{0, 0.354, 0.708, 1.062\}$
- $\tan^{-1}(x)$; $x_i = \{0, 0.697, 1.425, 2.672, 5.617, 16.963\}$

3.4 Algorithmic Description of the PWL Approximation

The stages in the approximation process are: (i) modification of the saturation PWL original saturation as defined in (2) to adopt it to every interval obtained previously and (ii) superposition of these modified saturation in order to obtain $\overline{f}(x)$.

The original saturation function $y(x)$ is modified by the affine transformation given by $\frac{1}{b_i}y(a_i x - c_i) + m_i$. This transformation translates the corners from (-1,-1) and (1,1) to $(c_i - \frac{1}{a_i}, m_i - \frac{1}{b_i})$, $(c_i + \frac{1}{a_i}, m_i + \frac{1}{b_i})$. This transformation is performed on the CNN by means of the following two templates run in a sequential way $TEM_{PWL^k,1} = \left\{A_{ij}^2 = 0; B_{ij}^1 = a_k\delta_{2j}\delta_{2i}); I^k = -a_k c_k \ ,\forall i,j\right\}$, $TEM_{PWL^k,2} = \left\{A_{ij}^2 = 0; B_{ij}^1 = b_k^{-1}\delta_{2j}\delta_{2i}); I^k = m_k \ ,\forall i,j\right\}$.

The way to obtain the desired approximating function is by shaping the output saturation by templates given by $TEM_{PWL^k,1}$ and $TEM_{PWL^k,2}$ with the coeficients obtained in the manifold stages needed to approximate the function.

[2] The reason for selecting these intervals will be shown in the following section.

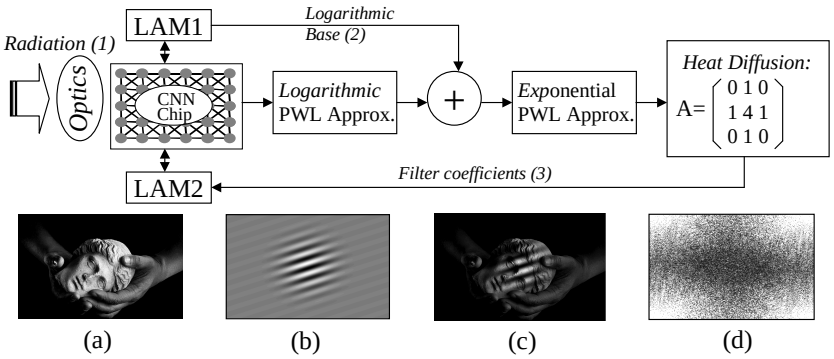

Fig. 3. Algorithmic diagram of the Integral Transformation.

Once we have modified the first stage, we apply this shaped output to the input image and save the result into the LAM, then the second stage is modified and the same input image is passed through this saturation and the results is added to the value previously saved in the LAM. Making this process through every stage we finally obtain the image processed by a point operation that approximately performs a desired function $f(x)$.

4 Gabor Transform Computation

This Section describes the design of a CNNUM program which compute the output of integral transforms. The base of the algorithm is described in Fig. 3. In this figure can be seen how the input image is directly irradiated on the Chip surface through the optical devices and sensed by the photosensors implemented on the silicon. Once we have the image in the LAM $I(x, y)$, we perform the approximation of the logarithmic function $\overline{\ln}(I(x, y) + 1.5)$, adding a bias on the whole image to translate the swing of analog values into the interval $(0.5, 2.5)$ where the logarithmic approximation is valid. Then we add in the LAM the logarithmically shaped image with the logarithmic base $B_{\log}(x, y) := \ln(B(x, y) + 1.5)$, where B(x,y) is the transformation base used. Then we pass this results through an exponential approximation stage to obtain

$$\begin{aligned} &\overline{exp}\left[\overline{\ln}(I(x,y)+1.5)+\ln(B(x,y)+1.5)\right] \\ &\quad = (I(x,y)+1.5)\,(B(x,y)+1.5) \\ &\quad = I(x,y)B(x,y)+1.5(I(x,y)+B(x,y))+1.5^2 \end{aligned}$$

It is easy to correct this result to obtain the product by simply substracting the term $1.5(I(x, y) + B(x, y)) + 1.5^2$ which is computing scaling the addition of input and base including a biasing term. Lastly, we perform the averaging by means of a template that emulated the PDE of temperature diffusion. This template gives us the value of the correlation between the image and the base

and the value of the transform at the frequency point of the base is directly calculated.

5 Experimental Results

In this section, it is presented a Gabor transform example providing run-time of the algorithm. Gabor filters has been used as preprocessors for different tasks in computer vision and image processing [6]. Gabor filters exists or signal of arbitrary dimension where an n-dimensional signal is defined to be a mapping from $\mathbb{R}^n$ to $\mathbb{R}$ or $\mathbb{C}$. For n-dimensional signals, the impulse response $g(\overrightarrow{x})$ of a two-dimensional Gabor filter is a complex exponential modulated by an n-dimensional Gaussian

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x^2+y^2)/2\sigma^2} e^{j(\omega_{xo}x+\omega_{yo}y)}$$

which is tuned to spatial frequency $(\omega_{xo}, \omega_{yo})$. This filter responds maximally to edges which are oriented at an angle $\theta = \tan^{-1}(\omega_{yo}/\omega_{xo})$ where θ is defined to be an angle between the horizontal axis and the line perpendicular to the edge.

Table 1. Comparision table of execution times of the different imaging processors (grey scale and 64×64 image resolution).

Run-time of elementary steps			Sub-tasks of the algorithm	
	CNN 100*nsec*	*PC* @4, 1*GHz*		*CNN* 100*nsec*
I/O image	6 μsec	10 μsec	*Logarithmic approximation*	24,4 μsec
Conversion LLM/LAM	100 nsec		*Exponential approximation*	61 μsec
Binary Save/Load	6 μsec		*Biasing*	6,1 μsec
Arithmetic operation	100 nsec	9 μsec	*PDE (Heat diffusion)*	8 μsec
Logical	50 nsec	8 μsec	*Image LAM transfer*	6 μsec
Convolution 3×3	1,4 μsec	32 μsec	*Coefficient LAM transfer*	6 μsec

Several illustrative images obtained along the transformation process are showed in Fig. 3. In table 1, we compare this architecture considering the processing time of different elementary steps in CNN and digital technologies. In our comparison we define equivalent operations between the parallel CNN and the serial processors.

6 Conclusions

In this paper, it has been introduced the equations that govern the complex CNN spatio-temporal dynamics and the CNNUM computational infrastructure implemented on silicon. After this, we have presented a general technique that allows us to approximate any nonlinear output function on the CNNUM VLSI Chip. For this purpose, we have given a theoretical analysis of an approximation technique based on the infinity norm criterion. Also, it has been comment the advantages of this technique in comparison with the cuadratic error criterion. Lastly, we have use this technique to implement the algorithm on the fully parallel architecture in order to implement Gabor filters. The main motivation of this work is to release CNNUM emphanalogic (analog+logic) architecture from using emphdigital computers when CNN image processing computing capabilities are unable to perform any required nonlinear filtering step or global transformation.

References

1. K.R. Crounse, and L.O. Chua, "Methods for image processing and pattern formation in cellular neural networks: A tutorial," *IEEE Trans. Circuits Syst.,* Vol. 42, no. 10, 1995.
2. L. O. Chua, and L. Yang. "Cellular neural networks: Theory". *IEEE Trans. Circuits Syst.*,vol. 35, no. 10. pp. 1257-1272, 1988.
3. L.O. Chua, and R.L.P.Ying, "Canonical piecewise-linear analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 125-140, 1983.
4. J.N. Lin, H. Xu, and R. Unbehahuen, "A generalization of canonical piecewise-linear functions," *IEEE Trans. Circuits Syst.*, vol. 41, no. 1, pp. 345-347, 1994.
5. G. Lin, S. Espejo, R. Domnguez-Castro, and A. Rodrguez-Vzquez, "The CNNUC3: an Analog I/O 64 x 64 CNN Universal Machine with 7-bit Analog Accuracy", IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA'2000), pp. 201-206.
6. S. Marceljia, "Mathematical description of the responses of simple cortical cells," *J. Opt. Soc. Amer.*, vol. 70, pp.1297-1300, 1980.
7. V.M. Preciado, "Real-time wavelet transform for image processing on the cellular neural network universal machine", *Lecture Notes in Computer Science*, Vol. 2085, pp. 636-643. Springer-Verlag, 2001.
8. T. Roska, A. Zarandy, S. Zold, P. Foldesy, and P. Szolgay, "The computational infrastructure of the analogic CNN computing - Part I: The CNN-UM Chip prototyping system", *IEEE Trans. Circuits Syst.*, vol. 46, no. 1, pp. 261-268, 1999.
9. P.L.Venetianer, F.Werblin, T.Roska, and L.O.Chua, "Analogic CNN algorithms for some image compression and restoration tasks", *IEEE Trans. Circuits Syst.*, vol. 42, no. 5, 1995.
10. F. Werblin, T. Roska, and L. O. Chua," The analogic cellular neural network as a bionic eye," *Int. J. circuit Theory Appl.*, vol. 23, no. 6, pp. 541-549, 1995.