Filtering and spectral processing of 1-D signals using cellular neural networks
Moreira-Tamayo, O.; Pineda de Gyvez, J.

Published in:

DOI:
10.1109/ISCAS.1996.541484

Published: 01/01/1996

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):
FILTERING AND SPECTRAL PROCESSING OF 1-D SIGNALS USING CELLULAR NEURAL NETWORKS

Oscar Moreira-Tamayo, and José Pineda de Gyvez
Department of Electrical Engineering
Texas A&M University, College Station, Texas 77843-3128 USA
Email: moreira@eesun1.tamu.edu, gyvez@pineda.tamu.edu, URL: http://silicon.tamu.edu

ABSTRACT
This paper presents Cellular Neural Networks (CNN) [1] for one-dimensional discrete signal processing. Although CNN has been extensively used in image processing applications, little has been done for 1-Dimensional signal processing. We propose a novel CNN architecture to carry out these tasks. This architecture consists of a shift register, e.g., a charge coupled device, and a lxn neural array. Each cell processes a sample of the input signal. By using appropriate templates and shifting the input signal the CNN array is capable of performing FIR filtering, discrete Fourier transform, and wavelet decomposition and reconstruction. Eventhough this implementation is not more efficient than conventional methods, the paper shows that an analog computer based on the CNN paradigm [2] can also be used to perform the linear operations described above. Simulation results and comparisons for spectral audio applications are presented.

1. INTRODUCTION
Several conventional architectures for CNN have been proposed [3,4]. In general, a CNN consists of an array of cells, each one connected only to its n-nearest neighbor. In previous architectures this array is 2-dimensional, and therefore intended mainly for image processing. Our approach consists of a 1-dimensional array. The CNN is operated by interacting with a memory which allows to input and output data. Usually, in practice, 1-dimensional signals are very long sequences compared with images. Therefore, to allow easy flow of data to and from our system we propose a memory unit that allows shifting the data along the array. This can be implemented with a charged coupled device or with a second layer of a 1-D CNN array. Notation and background definition are stated as follows:

The basic circuit unit of CNN is called a cell[1]. It contains linear and nonlinear circuit elements. Any cell, C(j), is connected only to its n-nearest neighbor cells. Such array is said to have radius n. Fig. 1 shows a radius 3 array. This intuitive concept is called neighborhood and is denoted as N(j). Each cell has a state x, input u, and output y. The state of each cell is bounded for all time t>0 and, after the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. This last fact is relevant because it implies that the circuit will not oscillate. The dynamics of a CNN have both output feedback (A) and input control (B) mechanisms. Notice that a 2-D CNN array of radius 1 has the same number of connections as a 1-D CNN array of radius 4. Therefore, their implementation complexity is similar. The shift memory shown in Fig. 1 is used for interaction of the CNN with the input data. Its main characteristic is its capability to shift its data through the memory locations. This allows to feed the input data to the memory as it is being sampled from an analog source. The implementation of such memory can be done with charged coupled devices as in [5]. The first order nonlinear

---

1 This research is partially supported by the Office of Naval Research under contract grant number N00014-91-1-0516

0-7803-3073-0/96/$5.00 ©1996 IEEE 76
differential equation defining the dynamics of a cellular neural network cell can be written as follows

\[
\frac{d x_j(t)}{dt} = - \frac{1}{R_1} x_j(t) + \sum_{C(k) \in N(j)} A(j;k)y_k(t) + \sum_{C(k) \in N(j)} B(j;k)u(k) + I
\]

\[
y_j(t) = \frac{1}{2} \left[ x_j(t) + I - |x_j(t) - I| \right]
\]

where \( x_j \) is the state of cell \( C(j) \), \( x_j(0) \) the initial condition of the cell, \( C_1 \) is a linear capacitor, \( R_1 \) is a linear resistor, \( I \) is an independent current source, \( A(j;k)y_k \) and \( B(j;k)u_k(t) \) are voltage controlled current sources for all cells \( C(k) \) in the neighborhood \( N(j) \) of cell \( C(j) \), and \( y_j \) represents the output equation. The input data is being fed to the memory from its first location and shifted along. When this data is being fed to the memory the CNN is set inactive. When the CNN is active the memory holds its values. Therefore \( u(t) \) remains constant during each processing period, i.e. \( u(t) = u_k \).

The operations to be implemented with this 1-D CNN array are the following:

- Causal Nth order FIR filter [6]:

\[
H(z) = \sum_{n=0}^{N} h(n) z^{-n}, \quad h(N) \neq 0.
\]

- Discrete Fourier Transform [6]:

\[
F(e^{j\omega_k}) = \sum_{n=-\infty}^{\infty} f_n e^{-j\omega_k n}
\]

However, since this function is periodic we only need to calculate one period \( n = 0..N/2\pi \). Therefore the function to implement is

\[
F(e^{j\omega_k}) = \sum_{n=0}^{N} f_n e^{-j2\pi n} = \sum_{n=0}^{N} f_n \left[ \cos(2\pi n) + j \sin(2\pi n) \right]
\]

where \( N \) is the number of points, and \( k=1..N \).

Decomposition for wavelets such as Daubechies' can be implemented using an FIR filter and the structure shown in Fig. 2 [7].

As we can see, the above operations are linear and discrete. Therefore, the CNN array needs to be operated in a discrete and linear fashion. A discrete-Time CNN has been previously proposed by Harrer et. al. [8]. The CNN can be used to perform linear algorithms by operating the network under the following conditions:

1. Set the bias current \( I = 0 \).
2. Set template \( A \) to zero.
3. Initialize the network to zero.
4. Let the network run for a fixed amount of time. That is, instead of letting the network run until it has converged, allow the network to run for a small amount of time \( 0 < \Delta t < 4\tau \), where \( \tau = R_1 C_1 \), and take the output from the state of the cell.
5. Take the output from the state of the cell. This will avoid the nonlinearity introduced by the limiter.

By using the previous conditions, equation (1) becomes:

\[
\frac{d x_j(t)}{dt} = - \frac{1}{R_1} x_j(t) + \sum_{C(k) \in N(j)} B(j;k)u_k(t)
\]

solving for the state of the cell for a time increment \( \Delta t \):

\[
x_j(t_0 + \Delta t) = x_j(t_0) + R_1 \frac{\Delta t}{R_1 C_1} \sum_{C(k) \in N(j)} B(j;k)u_k dt
\]

Consider \( x_j \) for \( t_0 = 0 \), where \( t_0 \) is the initial time. Then, we obtain

\[
x_j(\Delta t) = R \sum_{C(k) \in N(j)} B(j;k)u_k \frac{\Delta t}{R_1 C_1}
\]

Here we have a linear operation. Eq. 7 can be simplified assuming \( \Delta t = R_1 C_1 \).

Notice that contrary to a conventional linear operator in which the summation index runs for all the sequence, in
CNN the summation is performed only in the neighborhood of each cell. An additional constraint is that in a CNN array the template $B$ is set equal for all cells. This differs from a conventional linear operator in which the coefficients can be set independently for each input element.

2. **FIR Filter and Wavelet Transform Implementation**

An FIR filter is basically composed of multipliers and unity delays. Suppose that an FIR filter with coefficients $h(0), h(1), \ldots, h(N)$ is to be implemented with a 1-D CNN array consisting of $J$ cells and radius $R$ with $J = p(2R + 1)$ $N = qJ; \ p, q \in \mathbb{Z}$ (for feasible implementations $1 \leq R \leq 4$). This constraint is not required, but it simplifies the algorithm since it avoids exceptions at the ends of a sequence. Then, we can implement the FIR filter for any input vector of size greater or equal to $N$ with the following algorithm:

1. Initialize the CNN array to zero.
2. Set $I = 0$ and $A = 0$
3. Set $B = [h(0), h(1), \ldots, h(\infty)]$
4. Run the network for $\Delta t = R_1 C_1$
5. Shift the input data $R$ positions
6. Set $B = [h(R+1), h(2R)]$ and iterate from step 3 until all filter coefficients are used.
7. The output can be taken from any cell having a full set of connections (those in the edges have non symmetric and smaller neighborhoods). The cells that have a full set of connections contain the output with different delays. The first one contains the minimum delay and the last one the maximum delay that the network provides.

For example, suppose a CNN array of 14 cells and radius 3, and an FIR filter of 6th order with the following coefficients that correspond to a lowpass filter and $\omega_b = 0.1$ ($\omega = 1$ is half the sampling frequency)

$$h = [0.0212, 0.0897, 0.2343, 0.3094, 0.2343, 0.0897, 0.0212]$$

In this case the template $B$ would be set equal to $h$. After the first 7 points are processed, the memory is shifted seven positions. Cells $C(4)$ to $C(11)$ contain the output of 7 consecutive points. Cells $C(1)$ to $C(3)$ and $C(12)$ to $C(14)$ contain incomplete outputs since they do not contain all the input information due to their proximity to the end of the array. They are called border cells.

The Daubechies' wavelet transform can be implemented with this procedure. Two filters are implemented, a low pass and a high pass (see fig. 2). For a 4 point Daubechies wavelet the coefficients are:

$$\phi = [d_0, d_1, d_2, d_3], \ \text{and} \ \psi = [d_3 - d_2, d_1 - d_0], \ \text{where}$$

$$d_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}}, \ d_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, \ d_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}}, \ d_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}$$

If a 1-D CNN of radius 1 is chosen, template $B$ cannot contain all four coefficients. The solution process can be divided in two iterations with two templates, one for each iteration (steps 3 to 6). To calculate a coefficient, the templates for low pass would be $B_1 = [d_1, d_2, 0]$ and $B_2 = [d_3, d_4, 0]$ and for the high pass $B_1 = [d_1, -d_2, 0]$ and $B_2 = [d_3, -d_4, 0]$. It can be seen that the functions $\phi$ and $\psi$ are included in $B_1$ and $B_2$.

The signal can be further decomposed in more frequency bands by downsampling the data and using the low pass output as input and then processed again.

3. **Discrete Fourier Transform Implementation**

A Fourier transform is a complex function. Real and imaginary parts are computed separately in the CNN array. For either the real or the imaginary parts, the cosine or sine coefficients (see Eq. 4), are calculated beforehand and provided in template $B$ similarly as in the FIR filter. This computation can be quite extensive. For a 512 point DFT the number of coefficients $N$ is 512 for each $\omega_k$. This high number of coefficients can be significantly reduced when $N$ is a power of 2 because many coefficients will repeat and can be reused. The DFT algorithm is the following:

1. Initialize the CNN array to zero.
2. Set $I = 0$ and $A = 0$
3. Set $B = [\cos(0\omega_k), \cos(1\omega_k), \ldots, \cos(R\omega_k)]$
4. Run the network for $\Delta t = R_1 C_1$
5. Shift the input data $R$ positions
6. Set $B = [\cos((\omega+1)\omega_k), \ldots, \cos((2R)\omega_k)]$ and iterate from step 5 until $N$ coefficients are used.
7. Iterate from step 3 for the next $\omega_k, i.e., \omega_k = 2\pi k/N, \text{for } k = 0, \ldots, N$.

4. **Noise Reduction Application**

Spectral processing can be used for noise reduction. Music and speech signals contain a sum of locally periodic signals. The frequency components of noise such as white, pink or brown are distributed along the spectrum. Therefore if the signal is broken in small segments, and leaving only those frequency components with higher energy, the noise can be significantly reduced. This can be
done by using short time Fourier transform. The results can be significantly improved by using a multiresolution scheme such as wavelet decomposition. This scheme gives the same weight to signals of different frequencies while Fourier transform techniques have a fixed time duration which gives more resolution to higher frequencies.

The following results were obtained by simulating the CNN analog computing paradigm previously described. They consist of a wavelet decomposition of an audio signal, the elimination of small frequency components using DFT on the subbands, and reconstruction of the signal. Fig. 3 shows the original signal. Fig. 4 shows the original signal with white noise added. These signals consist of 16384 points and were sampled at 8000 samples per second. To reduce the noise the input signal was decomposed using the Daubechies wavelet as mentioned in section 2. The smallest band consisted of 256 samples. On each subband the spectrum of the signal was obtained using DFT of 128 points and thresholded to a level of 1/4 the main frequency component on each segment.

5. CONCLUSIONS

A CNN architecture for the processing of 1-D signals has been proposed. The algorithms for filtering and spectral processing have been developed. The system was tested by using it in a noise reduction application, obtaining satisfactory results.

REFERENCES


