A universal interface between PC and neural network hardware

Citation for published version (APA):

Fong, A. Q., Kanji, A., Sanchez-Sinencio, E., & Pineda de Gyvez, J. (1995). A universal interface between PC and neural network hardware. In *Proceedings of the 1995 IEEE International Symposium on Circuits and Systems, 1995, ISCAS '95, 30 April - 3 May 1995, Seattle, Wisconsin*, (Vol. 2, pp. 1169-1172). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ISCAS.1995.520352>

DOI:

[10.1109/ISCAS.1995.520352](https://doi.org/10.1109/ISCAS.1995.520352)

Document status and date:

Published: 01/01/1995

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 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](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A Universal Interface Between PC and Neural Networks Hardware¹

Apollo Q. Fong, A. Kanji, E. Sánchez-Sinencio, and J. Pineda de Gyvez

Department of Electrical Engineering, Texas A&M University, College Station, Texas 77843-3128, USA

Abstract. A unified environment consisting of software simulation, hardware implementation, and an interface between a PC and the hardware of the cellular neural nets is presented.

I. INTRODUCTION

In the research and development of neural network circuits, tools are necessary for effective simulation, instrumentation, and measurement. In particular, image-processing applications using cellular neural networks [1]-[4] need dedicated hardware and software for graphics-intensive I/O. The user should be able to easily enter and modify data for experimentation. Control commands are also needed to initialize, process, and collect results from an expandable array of cells. Finally, the final results should be presented to the researcher in a meaningful form.

A software/hardware combination is proposed here as a solution to this problem, as shown in Figure 1. It

Fig. 1. Complete hardware setup

consists of a PC linked to a microcontroller interface board, which can be attached to different neural network configurations. The controlling software uses a Windows™ graphical user interface (GUI), enabling the creation of custom software that is both intuitive and easy-to-use. By adding a high level of abstraction between software and hardware, the user is shielded from the complexity of the neural network implementation's details. The transfer of data by the PC-CNN interface is carried out via an RS-232 serial link. The interfacing hardware is capable of converting data input by the user into analog voltages, which can be distributed to any node of the CNN. Results from the network are converted to binary numbers and relayed back to the PC software for display as a bitmapped image. Though a CNN image-

processing hardware is presented in this paper, other network topologies will work as well. The CNN simulator for PC's follows the theory reported elsewhere and some convergence criterion of [2].

II. CNN SOFTWARE AND GRAPHICAL USER INTERFACE

The Windows™ application software (written in pure C code) provides the user an easy way to control the CNN hardware, without having to deal with low-level system routines. Both hardware and software CNN processing options are provided to allow the comparison of results. This is particularly useful for benchmarking the accuracy of neural network hardware during its development process. Figure 2 illustrates the various menu options.

Fig. 2. Software interface showing a loaded bitmap image

The File menu allows the user to load standard .BMP bitmap images for processing. After images are processed, they can be saved to the hard-drive or to a floppy disk. Similarly, the CNN templates can be loaded and saved, using files with the .TEM extension.

The Edit menu of the PC-CNN simulator contains selections for the pre-processing of images and templates. The template editor, shown in Figure 3, allows the modification of the A and B templates [1]-[2], the input bias, and the initial conditions. An auto-save feature can be set to automatically save templates

1. This work was partially supported by the Office of Naval Research under contact grant number N00014-91-1-0516

Fig. 3. The template editor of the PC-CNN simulator

after being edited. An Add Noise option randomly scatters black and white pixels throughout the input image to allow testing of noise-removal templates. This option can be repeatedly selected as desired without permanently changing the original bitmap file.

Clicking on the Process menu initiates the CNN array processing. An hourglass cursor indicates that the user must wait for a duration of time, which depends on the size of the loaded image. The program can also detect the absence of the neural network hardware, and will alert the user immediately after an attempt at hardware processing.

An Options menu enables the user to configure the processing and I/O settings. The Image Type option contains choices for black-and-white or color processing (the latter is for future development). The Process Method option sets the program's mode for either hardware processing or software emulation. As mentioned earlier, this feature is good for comparing hardware processing with known software results. Serial port settings (COM port, baud rate, parity, etc.) can be configured to work with most PC-compatible computers.

After an input image and template is loaded, processing the image is as simple as selecting the Process menu option. All necessary information is sent to the CNN, and is returned when completed. Finally, an output window quickly appears to reveal the newly processed image.

III. SERIAL COMMUNICATIONS PROTOCOL

The transfer of data between the PC and interfacing hardware uses an error-detection protocol that has three levels of detection. This was developed to prevent possible noise during transmission, which can alter the processed image and give misleading

results. This can be a problem, especially when using noise-removal templates.

The first method of detection uses bit parity-checking that can detect simple errors in the data bits. The parity can be either even or odd, as defined in the RS-232 standard. Since this will only detect bytes that were changed, additional methods were employed to detect missing bytes. The second method uses frequent handshaking signal exchanges to confirm that all data has been received. The PC and interface board are always kept in constant communication, enabling each to know what the other is doing. The third method is a "time-out" feature that aborts the transfer after a certain period of inactivity. This prevents the possibility of errors caused by corrupted handshaking signals, as well as disconnected hardware.

In Figure 4, the protocol flowchart of the PC-to-CNN transfer is shown. The interface board initially waits in a loop, polling for a *SEND* request from the PC. The board responds with an *OK* signal to confirm that the command has been received. The templates, images, and initial conditions are sent in packets to keep all data synchronized. This accomplished by beginning and ending each packet with *START* and *END* signals. This prevents missing data from being incorrectly shifted into subsequent data packets. Any time one of these handshakes are not fully completed, the protocol aborts the transfer and waits for the *SEND* command again. This will also detect any corrupted signals. For the CNN-to-PC direction of transfer, the protocol is reversed in a similar manner.

Fig. 4. PC-to-CNN communications protocol flowchart

IV. CNN CELL HARDWARE

A time-multiplexing CNN scheme [2], shown in Figure 5, was used to limit the amount required hardware. It consists of a 4x4 array of identical cells that collectively perform analog neural processing.

Fig. 5. Time-multiplexing a large image

Each cell is attached to its own small motherboard which is joined to other motherboards via ribbon cable. Figure 6 illustrates how the symmetry of the motherboard's open-architecture design cleverly routes neighboring signals to their correct destination. For the case of signals between diagonal cells, each motherboard uses adjacent boards as stepping-stones to carry the signal diagonally. This makes the task of expanding the network as simple as "plugging-in" more cells. Each CNN cell consists of eighteen analog multipliers, an opamp, and an analog multiplexer, shown in Figure 7. Future integrated circuit CNN

Fig. 6. Modular cell design

Fig. 7. CNN cell block diagram

hardware implementations can be used without changing anything else.

The multipliers are used to multiply the weights of the **A** and **B** templates with the feedback **y** and input **u**, respectively. These results are sent through a lossy RC integrator that controls the neuron's convergence time. The opamp is used as the thresholding activation function

$$y_{ij} = f(x_{ij})$$

where x_{ij} is the current state of cell (i, j) . There are four modes of operation that is controlled by the switching of the multiplexer. The first initializes the state of all cells to their corresponding input pixel value, at the time-multiplexing window's present position within the image. The second mode initializes all cells to a user-defined global state, which can be either black or white. The third mode switches-in the integrators for all of the cells to begin the neural processing. The final mode disconnects the interactions among cells to hold their final converged values for retrieval.

A global voltage bias is sent to all of the cells to provide an adjustable offset that is part of the user-defined templates (i.e. edge detection, hole filler, etc.). This is used instead of a current bias because the summing of all neighboring interactions are performed in the voltage domain. Once the processing is complete, the analog voltage outputs of each cell are converted to a binary numbers by the CNN interfacing hardware.

Although the hardware can currently process black-and-white, a provision has been designed to easily upgrade the circuit to handle color images. The output **y** would simply be replaced by the state **x**, to allow intermediate voltages to simulate varying pixel intensities.

V. INTERFACING HARDWARE

In order for the CNN hardware to receive pixel values, special circuitry is needed to latch and hold data sent by the PC. It must also provide a way to convert and return the image to be displayed.

The interfacing hardware consists of a 16-bit Motorola HC16 microcontroller attached to decoding circuitry, as in Figure 8. The HC16 data bus is decoded into 60 unique select lines, 56 of which are used to initialize individual sample-and-hold chips. A data bus latch provides control lines to select the CNN's mode of operation, as described earlier. The sample-and-hold chips contain voltage values that must remain constant for the duration of each

Fig. 8. Interfacing hardware diagram

time-multiplex: 1) template **A**, 2) template **B**, 3) V_{Bias} , 4) input image **u**, and 5) the voltage V_{Global} . A static border around the multiplex window is also assigned pixels from the input image to assist in the overlapping of multiplexes, minimizing the error caused by missing neighbor cells. Each line must be selected one-at-a-time to share the HC16's single 16-bit D/A converter. Analog representations of the required constant voltage values are output and stored in the sample-and-hold chips for the CNN hardware to access.

There are sixteen 4-to-1 multiplexers, which can select cell output voltages in groups of eight back to the HC16's 8-channel, 8-bit A/D converter.

The microcontroller software is stored in two 32k EPROMs, and is used to manage the time-multiplexed processing and to communicate with the PC. This serial link is established using the standard RS-232 cable, communicating at a speed of 19,200 bits-per-second (bps). After the PC sends the images, templates, initial conditions, the multiplexing routines sends the necessary data to the CNN through the D/A converter and waits until the cells converge. The A/D converter returns the converged CNN pixel values. The processed pixels are then stored in a separate address of memory, as the multiplex window scans across the entire original image. When the processing is complete, the microcontroller sends the resulting image to the PC using the serial transfer protocol.

VI. RESULTS

Software simulations were done on a black and white picture containing added noise, as shown in Figure 9a. Using a noise-removal template gives the result in Figure 9b, with all isolated white or black pixels removed. Figure 9c shows the application of the edge-detection template. The resulting image yields a traced outline of the original image.

Fig. 9. a.) Original image, b.) image after applying noise removal, and c.) after applying edge detection.

VII. CONCLUSION

An integrated system for image processing is proposed. The main elements of the system consist of a CNN software simulator, multiplexing hardware, and an interface between a PC and the hardware and software.

VIII. REFERENCES

- [1] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Trans. Circuits Systems*, vol. 35, pp.1257-1272, 1988.
- [2] C.C. Lee and J. Pineda de Gyvez, "Time-Multiplexing CNN Simulator," *Proc. of Intl. Symposium on Circuits and Systems*, pp. 407-410, May 1994.
- [3] J.A. Nossek and T. Roska, editors. "Special Issue on Cellular Neural Networks," *IEEE Trans. Circuits and Systems*, vol. 40, March 1993.
- [4] K. Halonen et al., "VLSI Implementation of a reconfigurable cellular neural network containing local logical," *Int. J. Circuit Theory Applications*, 1992.