RC25051 (W1010-013) October 6, 2010
Electrical Engineering

IBM Research Report

Locally Connected Processor Arrays for Matrix Multiplication and Linear Transforms

Chai Wah Wu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Locally Connected Processor Arrays for Matrix Multiplication and Linear Transforms

Chai Wah Wu
IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA
contact email: chaiwahwu@ieee.org

***Abstract*—Cellular Neural Networks is a multiprocessor computing architecture where the processors are only directly connected to nearby processors. This results in a trade off between the number of connections between processors and the number of steps needed to perform global computation. We consider such a locally connected computing architecture and present some preliminary analysis on this trade off and study this architecture's applicability to the specific problem of matrix multiplication including linear transforms applications such as 1-D and 2-D DCT and DWT. We illustrate that in general there is a trade-off between the following 3 parameters: the number of iterations needed to perform the global computation, the amount of memory in each processor and the connectedness of the graph. This latter parameter is expressed as the relative diameter of the computer architecture graph with respect to the problem graph.**

I. Introduction

Cellular Neural Networks (CNN) [1] is a multiprocessor computing architecture where the processors are only directly connected to nearby processors. Each processor contains compute units and memory. At the outset, each processor's contains data pertinent to the problem at hand. This is a particularly common arrangement in CNN vision-related applications where each processor is located near a sensor and processes the sensor data [2]. How the processors are connected plays an important role in the capabilities of the system. The locality of the processors' communication means that it cannot perform global computations that require knowledge of the data in all the processors, at least not in one step. However, one can trade off space connectivity with time. Over several iterations, data dissemination from processors to neighboring processors will be able to propagate information to all the processors allowing global computation to occur.

In this paper, we look at some aspects of this trade-off by studying a general locally connected computing architecture. In particular, the feasibility of using such an architecture for performing matrix multiplication will be investigated. As an example, we illustrate the possibility of using such an architecture for performing DCT and DWT.

The question of locality in distributed algorithms was studied in [3], [4] where the focus is on graph theoretical algorithms such as graph coloring and finding the maximal independent set. Our focus here is on signal and image processing applications that CNN is especially suitable for.

II. Notations and definitions

Definition 1 (Relative diameter): Given two graphs with the same vertex set V, the diameter of graph $G_A = (V, E_A)$ relative to graph $G_B = (V, E_B)$ is defined as $\max_{(v,w)\in E_B} \mathrm{dist}_{G_A}(v, w)$, where $\mathrm{dist}_{G_A}(v, w)$ is the distance between vertices v and w in graph G_A. We denote this as $\mathrm{diam}_{G_B}(G_A)$.

Note that the normal definition of diameter is the same as the diameter relative to the complete graph. Also note that the diameter of a connected graph relative to any graph is finite.

Definition 2: The r-neighborhood of a vertex v in graph G (denoted as $B_r(v, G)$) is defined as the set of vertices that is at most a distance r from v. We adopt the convention that $v \in B_r(v)$.

Definition 3: The matrix of a graph (V, E) of order n is an n by n matrix A such that $A_{ij} = 1$ if and only if $(v_i, v_j) \in E$ and 0 otherwise.

Note that the matrix of a graph depends on a specific labeling of the vertices to the integers $\{1, \cdots, n\}$.

Definition 4: The graph of a n by n matrix A is the graph on n vertices such as there is an edge between vertex i and vertex j if and only if $A_{ij} \neq 0$.

We consider the following locally connected computing architecture. A computer $(G, M, \{f_i\}, \{u_i\})$ consists of processors as vertices located on a graph $G = (V, E)$, where for each $i \in V$, the processor v_i contains M (local) memory locations m_i^j, $j = 1, 2, \ldots, M$. At each iteration t, each processor v_i computes a function using the data in its memory locations and the memory locations of its neighbors and stores them in the memory locations, overwriting the previous content, i.e. it computes $m_i^j(t) = f_i^j(\cup_{k\in B_1(v_i,G), 1\leq l\leq m} m_k^l(t-1), t)$ for each $1 \leq j \leq M$. The output of the computer is defined as $y_i(t) = u_i(\cup_j m_i^j(t))$ for each $i \in V$.

III. Trade-off between connectivity, time and memory

In a CNN architecture the processors are locally connected, i.e. the graph G is a locally connected graph. We would like to know what the trade-offs are in terms of connectivity of the graph, the time it takes to compute the result and the memory needed. The following simple result shows that it is possible to compute anything a more densely connected architecture can do, but at a cost of increased time and memory:

Theorem 1: Assume that G_A and G_B have the same vertex set V. Consider a computer $\mathcal{C}_B = (G_B, M_B, \{f_i^B\}, \{u_i^B\})$

computing $\{y_i^j\}$ in one step. Then for any graph G_A there exist a computer $\mathcal{C}_A = (G_A, M_A, \{f_i^A\}, \{u_i^A\})$ computing $\{y_i\}$ in d steps where $d = \text{diam}_{G_B}(G_A)$ and $M_A = |V|M_B$.[1]

Sketch of Proof: Since $M_A = |V|M_B$, each processor v in $\mathcal{C}_A$ has enough storage to store the data in all the processors of $\mathcal{C}_B$. For $1 \leq t < d$, let f_i^A be defined as the function that propagate the data in the memory to the memory of its neighbors in the appropriate slots. At iteration $d-1$, all the data in each processor or its neighbors in G_B is either in local memory or the memory of its neighbors in G_A. The d-th iteration will serve to compute the final result. □

This result shows that the number of iterations needed to mimic the other machine is d. The discussion about the propagation of data in the proof also shows that this is the minimal number of iterations needed; there exists computations for which at least $d-1$ steps are needed to propagate the data to all the necessary nodes.

However, the requirement on the memory of $\mathcal{C}_A$ in Theorem 1 is quite conservative and we like to know what the minimal amount of needed "extra" memory is. In the following section, we look at this issue for the problem of matrix multiplication which include linear transforms such as DCT and DWT.

IV. Matrix multiplication

Consider the case where each f_i computes vector weighted sums of data in the memory locations of the processor and its neighbors and store the result into the local memory. Thus each iteration corresponds to a matrix multiplication operation with the vector of data in memory.

For the simplest case where each processor has a scalar memory location (i.e. $M = 1$), the corresponding problem is the following: Given a matrix A and a graph G, find a set of d matrices A_i such that $\|A - \pi_i A_i\| \leq \delta$ for some $\delta > 0$ under the constraint is that the graph of each A_i is a subgraph of G.[2] What can we say about d, the number of matrices in the set $\{A_i\}$? It is easy to show that in general the number d needs to be at least $\text{dist}_{G_A}(G)$ where G_A is the graph of the matrix A. Furthermore, by counting the degree of freedom in A and A_i as the number of nonzero elements, we see that in general we also require d to be at least $\left\lceil \frac{n(A)}{n(G)} \right\rceil$ where $n(A)$ is the number of nonzero elements in A and $n(G)$ is the number of nonzero elements of the adjacency matrix of G. For a specific matrix A, this latter constraint on d might not be needed.

A. Discrete Cosine Transform (DCT)

Consider the n by n orthogonal matrix C corresponding to the 1-D DCT of size n: $C_{1j} = \frac{1}{\sqrt{n}} \cos\left(\frac{\pi}{n}\left(j - \frac{1}{2}\right)(i-1)\right)$ for $j = 1, \cdots, n$ and $C_{ij} = \sqrt{\frac{2}{n}} \cos\left(\frac{\pi}{n}\left(j - \frac{1}{2}\right)(i-1)\right)$ for $i = 2, \cdots, n$, $j = 1, \cdots, n$.

Consider a CNN topology where the processors are arranged in a line and each processor only communicates with its nearest neighbors. This means the the underlying graph is the path graph P_n. The graph of the matrix C is the complete graph K_n. To implement this on a path graph P_n topology would require at least $n-1$ iterations since the diameter of P_n is $n-1$.[3] Let us consider the case $n = 8$. For various values k, we use a line-search algorithm to find a set of k matrices A_i such that $\|C - \pi A_i\|$ is minimized with each of the graph of A_i a subgraph of P_8. The matrix norm used will be the Frobenius norm. Figure 2 shows the minimal value of this norm found for various values of k and various types of graphs. As can be seen, a transition occurs at $k = 7$ for the path graph, in accordance with the above conclusion that at least 7 matrices are needed to approximate C.

As an illustration, the following set of 7 matrices A_i when multiplied together generate a matrix $A_1 A_2 \cdots A_7$ that is close to the 1-D DCT matrix C of order 8. Each A_i is tridiagonal and thus has P_8 as its graph.

$$\begin{pmatrix} 1.162 & -0.802 & & & & & & \\ 0.573 & 0.887 & -1.250 & & & & & \\ & 0.719 & 1.273 & 1.399 & & & & \\ & & 1.328 & 0.480 & 0.462 & & & \\ & & & -1.027 & -0.418 & -0.490 & & \\ & & & & 0.888 & 0.044 & 1.478 & \\ & & & & & -1.540 & -0.466 & -0.039 \\ & & & & & & 0.969 & 1.029 \end{pmatrix}$$

$$\begin{pmatrix} 0.784 & 0.037 & & & & & & \\ 0.194 & -0.455 & 0.994 & & & & & \\ & 1.165 & 0.579 & 0.749 & & & & \\ & & 0.083 & 0.946 & -0.047 & & & \\ & & & -0.479 & 0.678 & -0.232 & & \\ & & & & 0.332 & 0.555 & 0.673 & \\ & & & & & -0.503 & 0.604 & 0.663 \\ & & & & & & 0.321 & -1.044 \end{pmatrix}$$

$$\begin{pmatrix} -0.301 & 0.591 & & & & & & \\ 0.746 & 0.899 & 0.237 & & & & & \\ & -0.123 & 1.227 & -1.257 & & & & \\ & & -0.242 & 1.027 & -0.469 & & & \\ & & & 0.219 & 0.058 & 0.768 & & \\ & & & & 0.672 & -0.070 & 0.073 & \\ & & & & & 0.129 & 0.129 & 1.212 \\ & & & & & & 1.295 & 0.110 \end{pmatrix}$$

$$\begin{pmatrix} -0.784 & 0.006 & & & & & & \\ 0.598 & -0.036 & 1.156 & & & & & \\ & 0.854 & -0.488 & -0.413 & & & & \\ & & -0.584 & 0.571 & -0.445 & & & \\ & & & 0.516 & 0.761 & -0.267 & & \\ & & & & -0.025 & 0.458 & 0.923 & \\ & & & & & -1.008 & 0.581 & -0.654 \\ & & & & & & -0.023 & -0.841 \end{pmatrix}$$

$$\begin{pmatrix} 0.798 & 0.735 & & & & & & \\ 0.719 & -0.516 & -0.292 & & & & & \\ & -0.001 & 0.389 & 1.251 & & & & \\ & & -0.674 & 0.142 & 0.305 & & & \\ & & & 0.666 & -0.405 & 0.425 & & \\ & & & & 0.410 & 0.971 & -0.194 & \\ & & & & & 0.386 & 0.891 & 0.570 \\ & & & & & & 0.681 & -0.362 \end{pmatrix}$$

$$\begin{pmatrix} 0.804 & -0.415 & & & & & & \\ 0.344 & 0.290 & 0.886 & & & & & \\ & 1.093 & -0.169 & 0.286 & & & & \\ & & 0.797 & -1.048 & 0.607 & & & \\ & & & 0.343 & 0.488 & 0.603 & & \\ & & & & 0.774 & -0.357 & 1.134 & \\ & & & & & 0.630 & 0.919 & -0.991 \\ & & & & & & 0.350 & 0.446 \end{pmatrix}$$

$$\begin{pmatrix} 1.409 & 0.758 & & & & & & \\ 0.783 & -1.612 & 0.867 & & & & & \\ & 0.446 & 0.914 & 0.352 & & & & \\ & & 1.364 & -0.404 & -0.037 & & & \\ & & & -0.760 & 0.767 & 0.191 & & \\ & & & & -0.536 & 1.141 & 1.023 & \\ & & & & & 0.796 & 0.016 & -1.644 \\ & & & & & & 0.954 & -0.270 \end{pmatrix}$$

[1] Assuming that d is finite.

[2] Since each processor can access its own local memory, we assume that all the graphs G we consider has self-loops added, i.e. there is an edge from each vertex to itself.

[3] One can see that the FFT butterfly network is equivalent to the case where the coupling graph can change at each iteration. At each iteration, each graph is a disconnected set of $n/2$ edges. Because of the time-varying nature, each node can connect to every other node in $\log_2(n)$ iterations.

Consider now processors arranged on a 2-D grid graph (Fig. 1). Since the 2-D DCT is separable into 1-D DCTs applied to the rows and columns sequentially, the result for the path graph can be applied here first row-wise and then column-wise and this shows that the 2-D DCT of order 64 can be computed on the grid graph topology in $7+7=14$ iterations. Thus we can implement an order 64 2-D DCT on a CNN architecture where each processor performs 3 multiply-and-add's in parallel and repeat for 14 iterations.

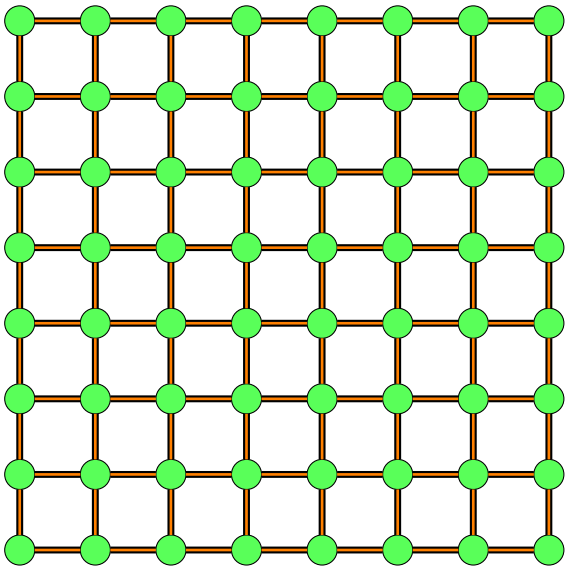

Fig. 1. 2-D 8 by 8 grid graph.

Figure 2 also shows that the star graph topology and the balanced binary tree[4] topology (Fig. 3) can approximate C using fewer matrices at $k=6$ and $k=5$ respectively.

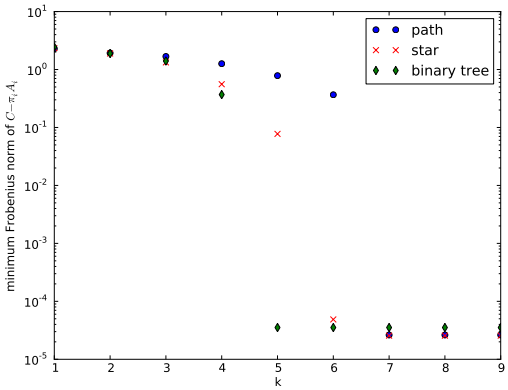

Fig. 2. Minimum Frobenius norm of $C-\pi_i A_i$ for various values of k and graph topologies.

This indicates that among the 3 types of spanning trees (path, star and balanced binary tree), the binary tree is a better

[4]defined as a binary tree with the smallest diameter.

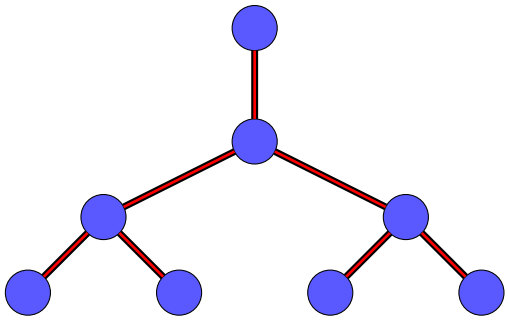

Fig. 3. Balanced binary tree of 8 vertices.

topology to compute the DCT. From the degree of freedom discussion earlier, a matrix whose graph is a spanning tree will have $8+7+7=22$ nonzero entries, and thus we expect to require at least $\lceil \frac{64}{22} \rceil = 3$ matrices A_i to form an adequate approximation of C. Numerical experiments indicate that among all spanning trees (which necessarily have 7 edges) at least 5 matrices are needed. On the other hand, by adding a single edge to the balanced binary tree in Fig. 3, we obtain the following graph of 8 edges (Fig. 4), for which only 4 matrices are needed.

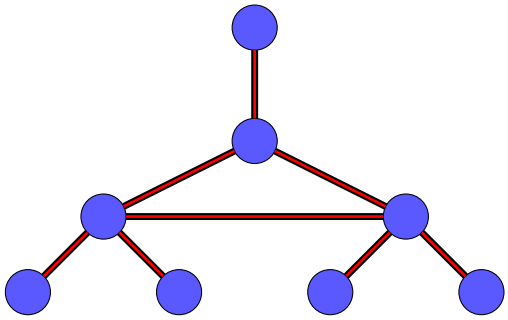

Fig. 4. A graph of 8 edges for which only 4 matrices are needed to approximate C.

How many edges does the graph need to have in order to require only 3 matrices to approximate C? It turns out a graph of 10 edges is sufficient. For instance, the graph in Fig. 5 obtained by adding 3 edges to Fig. 3 is such that only 3 matrices with this graph as the matrix graph are sufficient to approximate C. For instance the following 3 matrices when multiplied result in a matrix that is close to C.

$$\begin{pmatrix} 0.145 & & & & 0.584 & & 1.214 & -0.334 \\ & 0.145 & & & & 0.857 & -0.543 & -0.766 \\ & & -0.096 & & & 0.568 & 1.012 & \\ & & & 1.050 & & & & 0.616 \\ -1.175 & & & & 1.211 & & & \\ & 1.099 & -0.448 & & & -0.255 & & -0.922 \\ -0.070 & 0.262 & 0.940 & & & & 0.229 & \\ 0.671 & 0.783 & & -0.762 & & 0.296 & & -0.770 \end{pmatrix}$$

$$\begin{pmatrix} -0.717 & & & & 0.509 & & 0.206 & -0.355 \\ & -1.069 & & & & -1.820 & 1.182 & -0.953 \\ & & -0.845 & & & 0.179 & -0.390 & \\ & & & -0.976 & & & & -0.810 \\ -0.166 & & & & 1.166 & & & \\ & 0.539 & -0.611 & & & -0.862 & & -0.958 \\ 0.497 & -0.399 & 0.612 & & & & 0.511 & \\ -0.491 & -0.209 & & 0.833 & & -0.683 & & -0.275 \end{pmatrix}$$

$$\begin{pmatrix}
0.508 & & & & -0.919 & & -0.400 & 0.799 \\
 & 0.914 & & & & -1.008 & 0.182 & -0.207 \\
 & & -1.156 & & & -0.408 & 0.531 & \\
 & & & -1.148 & & & & 0.879 \\
0.268 & & & & 1.158 & & & \\
 & 0.769 & 1.165 & & & 1.198 & & 0.329 \\
-0.175 & 0.879 & 1.458 & & & & 0.401 & \\
-0.558 & -0.337 & & 0.849 & & -0.849 & & -0.380
\end{pmatrix}$$

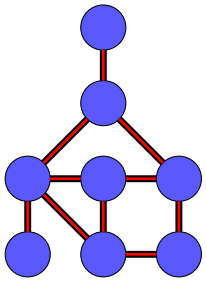

Fig. 5. A graph of 10 edges for which only 3 matrices are needed to approximate C. The diameter of this graph is 3.

B. Discrete Wavelet Transform (DWT)

The number of steps needed to compute a full matrix A using sparse matrices A_i depends on the entries of A. Let us study another widely use linear transform and compare it with the DCT. Consider the 1-D 3-stage Cohen-Daubechies-Feauveau 9/7 discrete wavelet transform of order 8 and apply the same procedure to construct a sequence of matrices whose product approximates the DWT matrix (denoted as W). In Table I we show the number of steps needed to approximate the DCT matrix C and the DWT matrix W for various graph topologies:

	diameter	DCT matrix C	DWT matrix W
Path graph P_8	7	7	7
Star graph	2	6	7
Balanced tree (Fig. 3)	4	5	5
Fig. 4	3	4	4
Fig. 5	3	3	3

TABLE I
NUMBER OF STEPS TO APPROXIMATE THE MATRICES C AND W FOR VARIOUS GRAPH TOPOLOGIES.

We see that approximating the DWT matrix is similar to the DCT case (with the exception of the star graph), although we noticed that the optimization algorithm seems to have more difficulty finding a solution in the DWT case, suggesting that the DWT is a "harder" transform than the DCT.

V. TRADE-OFF BETWEEN MEMORY AND NUMBER OF ITERATIONS

Recall that from Theorem 1 the global computation can be performed in d steps on a locally connected architecture with sufficient auxiliary memory storage. On the other hand, the numerical results above show that for the 2 graphs in Figs. 3-4, the number of steps required is 1 more than the diameter d. This is because the number of memory locations in each processor is 1, the same as the global problem of computing C and presumably there is not enough degrees of freedom for the sparse matrices to approximate C. If we increase the number of memory locations in each processor to 2, and have each processor compute a weighted sum of its own data and those of its neighbors, then numerical experiments show that we could approximate the matrix C in d steps, the minimal required.

Consider next the star graph. This graph has diameter 2. For $M = 1$, the number of steps required is 6. Figure 6 shows that as we add memory to each processor, the number of steps needed decreases until at $M = 5$ the number of steps needed is 2, which is the minimum dictated by the diameter.

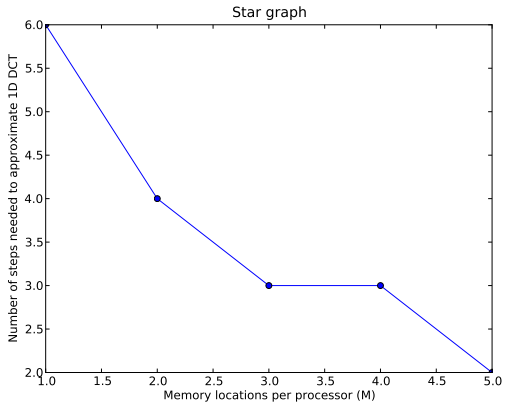

Fig. 6. Number of iterations needed to approximate C versus the number of memory locations in each processor for the star graph topology.

VI. CONCLUSIONS

We consider a locally connected computer architecture such as the CNN and study its feasibility in implementing global computations such as DCT and DWT. In particular, we show that for the 2-D DCT of order 64, processors located on a locally connected 2-D grid can approximate it in 14 iterations. In general, there is a trade-off between the following 3 parameters: the number of iterations, the amount of memory in each processor and the connectedness of the graph. This latter parameter is expressed as the relative diameter of the computer architecture graph with respect to the problem graph. More research is needed to determine the exact nature of this trade-off and perhaps the insights gained can be used to choose the optimal topology in a CNN architecture design.

REFERENCES

[1] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems-I*, vol. 40, pp. 147–156, mar 1993.

[2] B. J. Sheu, K.-B. Cho, and W. C. Young, "Integration of sensor/processor under cellular neural networks paradigm for multimedia applications," in *Proceedings of Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA)*, 1998, pp. 45–49.

[3] B. Awerbuch, M. Luby, A. Goldberg, and S. Plotkin, "Network decomposition and locality in distributed computation," in *Proceedings of 30th Annual Symposium on Foundations of Computer Science (FOCS)*, 1989, pp. 364–369.

[4] N. Linial, "Locality in distributed graph algorithms," *SIAM J. Computing*, vol. 21, no. 1, pp. 193–201, 1992.