COMPARING DIGITAL NEURAL NETWORK ARCHITECTURES

Yaron Kashai and Yair Be'ery *

Department of Electrical Engineering - Systems,
Tel-Aviv University, Tel-Aviv 69978, Israel.
* Also with the DSP Group Ltd, Givat-Shmuel 51905, Israel.

A method of assessing the quality of digital neural networks is proposed. This method, which consists of calculating a few measures, provides indications for network qualities, without specifying the processing elements. This feature makes the proposed method applicable to a variety of architectures. The method is demonstrated for three different architectures. The insight gained by measuring them suggested several improvements, whose contributions are predicted by the method.

INTRODUCTION

The goal of this paper is to define some measures that can be applied to a variety of adaptable, digital, synchronous silicon implementations of neural networks. These measures are intended as a tool for analysis of given implementations qualities, as well as a design aid for the development of new structures. They should reflect network model qualities such as speed, complexity and efficiency[1-3]. An attempt is made to make those measures as independent as possible. They must not reflect the technology used for implementation. Nor should they be affected by design decisions that can be easily modified.

The characteristics of neural network implementations are dictated by the performance of every processing element (PE) and by the inter-connections between them (the network). Our work concentrates on the network performance. This is useful because a large variety of networks can be compared, some of which have little in common in terms of PEs. It is also claimed that the PE design requirements are derived from the network design. In order to demonstrate the proposed measures, three networks [4-6] are analyzed along the paper. The networks have different network architectures and various PE implementations. Still, they are comparable.

The paper is organized as follows. The criteria for measurement are presented at the first part of the paper. The next three parts deal with deriving the characteristic figures from the three analyzed architectures. A comparative summary and a discussion are given in the last part.

CRITERIA FOR MEASUREMENTS

In this paper, the hardware implementation of the neural network is called "the network". The virtual neural network which is emulated by the hardware, is referred to as "the problem".

The following qualities are regarded as representative of a network model performance:

A) Convergence time, calculated by taking into account the typical bandwidth of the supporting interconnections.
B) Efficiency, which is the portion of processing elements that actually carry out calculations.
C) Complexity, which will reflect the amount of wires required for implementing the interconnections.

Convergence Time

Convergence time, marked here as τ, will tell about the raw power of the network. This measure ignores the cost of carrying out the calculation. It classifies networks according to how fast they can come up with an answer.

Three network styles where selected as test cases: a fully connected net, a feed forward layered net and a layered net using multiple back propagation [2]. In all models, when implemented on synchronous hardware, the network steps from one state to another, through some transient phase. The amount of time required by each transient phase is determined primarily by the interconnection bandwidth. The measurement of transient phase will serve as the speed measure. It will be counted in cycles, measured between two consequent activations of the update function.

The following parameters will be used for calculations:

N The actual number of neurons for the problems.
P Number of available processing elements.
K Number of layers for the layered cases.
M_i Size of layer i for the layered case. ($0<i\leq k$). M_1 will be the input layer, M_k will be the output layer.

Efficiency

Efficiency will indicate how much of the available computation power really goes into solving the problem. It is important to distinguish between PEs that actually calculate neuron states and those that do shifts or serve as registers. Processing elements used for registering or shifting are not doing "real" work. They are backing up a shortcome of the communication system. Such PEs will be counted out, in order to maintain the intuitive meaning of efficiency.

Efficiency is bound to change according to the network paradigm employed. Again, the fully connected case, the feed forward layered case and the back propagation layered network are used as benchmarks. Efficiency is also depended upon the relation between the network size and the problem size. Three cases should be considered: when the problem is smaller than the network (called here "under-utilization"), when the problem and the network are of equal sizes and when the problem is larger than the network (called here "over-utilization"). The derivation of measures for the under and over utilized cases is not described in detail, due to the limited scope. However, the results can be found in the summary.

Complexity

All the discussed models can be directly implemented. The complexity of wiring is of primary importance when dealing with micro-electronic implementation. It will be only fair to include a measure of wiring complexity, as a counter balance to all the measures affected by communication bandwidth. The basic entity suggested for measuring complexity, is a "track". A single bus connecting all processing elements is considered to occupy a track. Any number of busses can occupy the same track, as long as non of the processing elements is connected to more than a single bus. When two busses connect the same processor, they should be counted as two separate tracks. The direct implementation of a fully connected net of N neurons would cost N tracks. This is just the amount of communication that will be needed for a basic cycle in such a net, when implemented using a single track.

This measure is rather coarse, as it doesn't take into account the actual topology of the wiring. It also fails to account for interconnecting circuitry such as registers, switches and such. However it will reflect correctly the size limit imposed by the architecture. Parameters like die size and pin count may be used to refine the complexity measure. Complexity will be noted by χ.

NETWORK A

The network marked A is described in paper [4], *Linear Systolic Neural Network Engine.* Its basic architecture is of a single turoidal ring of PEs (See figure 1). Each PE is connected to its two direct neighbors. The basic communication operation involves a PE writing an output and its successor reading it. As all PEs act synchronously, the whole system can be regarded as a long shift register. Every PE has a local bank of synaptic weights. Every processing element represents a single neuron. The basic cycle of operation involves all PEs writing their state onto the shift register. A sequence of P–1 shifts expose every PE to the state of all

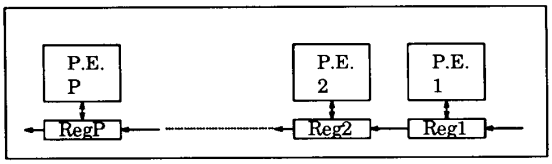

Figure 1. Network A connectivity.

other PEs, one at a time. During this sequence new activation values are accumulated for every PE. At the Pth step, every PE calculates its new state. For the layered cases, one must divide the PEs to K equal segments, each of the size of the maximal layer required. The calculation cycle is the same, with the exception that some connection weights are set to zero.

Calculating Convergence Time

The described form of operation suggests that a fully connected network consisting of N neurons will take N shifts, or basic communication operations to calculate the next state. This is under the assumption that exactly N PEs are used (N=P).

As mentioned above, the layered case can be considered as a fully connected net, with some of the connection weights set to 0. Feed forward nets can work in pipeline, if the learning paradigm is local to each layer. In that case, all layer values can be calculated during a full P step shift. Those values will represent K consequent steps of network states.

Back propagation is difficult to implement, due to the fact that every PE must sum the error feedbacks, multiplied by the synaptic weights between itself and all the PEs on the next layer. For a naive implementation, this will require maintaining the weight matrix twice. Once at the receiving layer, when the net is in the feed forward phase, and once at the sending layer, for back propagation. The way around this problem is described in [5]. Instead of propagating back the error value, each PE propagates the error value multiplied by the synaptic weight between itself and every PE on the preceding layer.

For network A, a full P–1 cycle shift is required to deliver error feedback. This full shift is required M_i times when layer i+1 feeds back to layer i. Thus

$$\tau = (K-1) * M * P \ ; \text{where } M = \max_{i=1}^{K}(M_i) .$$

(There can not be any pipelining during back propagation.) As the cost of back propagation is dominant over the cost of the feed forward stage, the second is neglected.

Efficiency Calculation

For the fully connected case, when P=N, all PEs carry out a calculation on every time step. This yields efficiency of η=1.

The efficiency calculation in the feed forward case will use the fully connected case as a base line. The difference here lays in the need to use layers in the size of the maximal one. This results in a number of leftover processors that do nothing except shifts throughout the process. The efficiency is therefore (with K stage pipelining):

$$\eta = \eta_{base} * \frac{N}{P_{req}} \ ; \text{where } P_{req} = K * \max_{i=0}^{K}(M_i) .$$

The efficiency in the back propagation case is low. Assuming that layer i is feeding back to layer i-1, each PE in layer i must calculate the multiplication of its error value with the weight of the connection to every PE on layer i-1. That gives M_{i-1} time steps, when only a single PE of level i is doing calculations. After every value is calculated, it must be communicated to layer i-1 for summation. Summing up the partial results should involve M_{i-1} PEs of level i-1, for a single step. The efficiency will be, therefore the ratio between the above calculation and the available number of PEs (P) multiplied by the total time required for the process. Namely,

$$\eta = \frac{\sum_{i=2}^{K} M_i * M_{i-1} * 2}{P^2 * (K-1) * M} \approx \frac{2*M}{P^2} .$$

Calculating Complexity

Network A has a single dimension of communication. On this dimension every PE has a single input and a single output. This is considered as a unit complexity measure, thus, according to the above definition: χ = 1 track.

NETWORK B

The idea behind network B is described in [5], *Implementing Digital Neural Network On A SIMD Processor*. Though the paper does not suggest direct implementation in hardware, the architecture can be used for that purpose. The basic architecture is a vector of PEs connected in line (See figure 2). In order to optimize the architecture for the multi-layer case, two channels of communication are provided. One for the inputs and one for the outputs. For the sake of the multi-layer back propagation, the communication shift registers are made bidirectional. That way error values can be delivered form one layer to its predecessor without involving other layers. The input and output shift registers can switch information by means of a special switch, located between every two PEs. Typically, those switches will be set at layer borders.

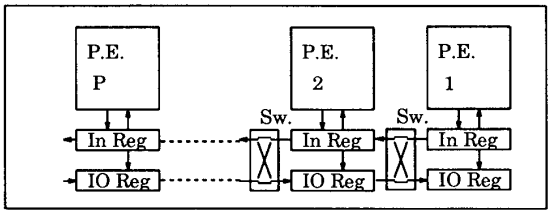

Figure 2. Network B connectivity.

Calculating Convergence Time

For fully connected nets, during the initial state of a basic cycle, every PE writes its current state to the output register. The output shift register must be connected externally to the input shift register. Trough this connection, all the current states are fed into the input shift register. Exposing the first PE to the states of all others would cost P–1 time steps. Additional P steps will be required to cover the distance to the last PE. At the 2Pth step the new state of each neuron is calculated.

According to the above, when P=N the convergence time will be $\tau = 2 * P$ time units.

In the feed forward case, the calculation time is depended upon the size of the input too. The notation M_0 will serve as the size of the input to layer 1, heron.

The computation of every layer i depends on the size of its input, M_{i-1}, and its own size (M_i). The computation for each layer terminates when the last input has passed the foremost PE. This will require $M_i + M_{i-1}$ time steps. The time required by the whole computation will be determined by the slowest layer.

$$\tau = \max_{i=1}^{K}(M_{i-1}) .$$

In the back propagation case too, network B is optimized for the task. Propagating the error from layer i to layer i–1 still involves M_{i-1} multiplications by every PE of layer i. Feeding back the intermediate results is much shorter for network B, involving only M_i additional shifts, at most. The accurate figure is

$$\tau = \sum_{i=2}^{K} \tau_i \ ; \text{where } \tau_i = \sum_{j=1}^{M_i} M_{i-1}+j .$$

Calculating Efficiency

It is already established that for the fully connected case N multiplications and summations are required, in every basic cycle.

Thus the general calculation of efficiency in this case will be

$$\eta = \frac{N}{\tau} .$$

This yields η=0.5 for the nominal case, where P=N.

The only idle time in the feed forward case is caused by the imposed delay between two consequent inputs to the net, as discussed above. If all layers are of the same size this case yields 100% efficiency. The time delay between two consequent inputs to the pipeline is τ. During this delay all layers are calculated once, causing PEs to be idle here and there. This yields an efficiency measure of

$$\eta = \frac{\sum_{i=1}^{K} M_{i-1} * M_i}{P^2} .$$

Back propagation, performed according to the previously used method is shown to employ a single PE for M_{i-1} time steps, and then M_{i-1} PEs for a single step, when feeding back from layer i to layer i–1. This process is repeated M_i times for that layer. Therefore the efficiency in this case is calculated as

$$\eta = \frac{\sum_{i=2}^{K} 2 * M_{i-1} * M_i}{P * \tau} = \frac{\sum_{i=2}^{K} 2 * M_{i-1} * M_i}{P * \sum_{i=2}^{K} M_i * (0.5 M_i + M_{i-1})} .$$

Calculating Complexity

Network B uses one dimensional communication. Two independent shift registers serve as communication channels. Therefore, the wiring complexity is $\chi = 2$.

NETWORK C

Network C is described in [6], *Customizable Neural Network On Silicon*. It is intended as a general purpose neural network. The PEs of network C are arranged in a matrix (See figure 3). On every border between rows there is a bus connected to PEs on both neighboring rows. These busses can be separated to segments, by soft-switches that are located at every column border.

The basic arrangement suggested by [6] is achieved by using the soft-switches to form a kind of a ring. However, as the architecture is very flexible, we took the liberty of suggesting different arrangements, for calculating the measures. Both arrangements yield the same results.

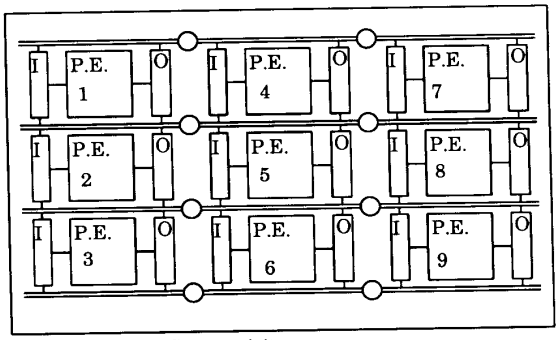

Figure 3. Network C connectivity.

Calculating Convergence Time

Using a fully connected setup, every PE has one bus segment that is shared with its predecessor and one that is shared with its successor. The first will be used for input, while the other will pass the output on. Under these conditions, network C performs exactly as network A. There is a single difference, however, in the case of under-utilization. Because the ring of network C is constructed using soft switches, one can tailor the network to the exact size, bypassing units that are not actually in use. This implies τ=N for the nominal case.

In a feed forward layered net, it is better to have separate shift registers for input and output, as demonstrated by comparing networks A and B. Network C can not be set up to form two shift registers. Instead, a single cyclic shift register will be used between layers. The shift register has the size of the number of inputs. It is loaded in a single step, in parallel. The values are dispensed by moving them around, so every PE gets to read them all. This process takes exactly the size of the input time steps.

Due to the restrictions in network C architecture the output of two consequent layers must be separated by at least one time step. Therefore the output stage will take two time steps. This method employs M*K PEs while a calculation step takes

$$\tau = M + 2 \text{ while } M = \max_{i=0}^{K}(M_i)$$

time steps for the nominal case.

In the back propagation case too, the preferred attitude is using two separate shift registers, one for input and one for output. Alternatively, a cyclic shift register is used between each two layers. Assuming the same implementation as in the layered case, the back propagation process involves M cycles of each shift register. Every PE of the propagating layer will use a single cycle of the shift register. During that cycle it will calculate the error value multiplied by the connection strength to each PE on the accepting layer. After M steps, the shift register has all the values, and all the PEs on the accepting layer read them in, during a single cycle. This process goes on for all the PEs of the propagating layer.

The process described above takes M^2 steps per layer. This is accumulative, as no pipelining is possible. This yields $\tau = K * M^2$.

Calculating Efficiency

Considering any of the arrangements mentioned for the fully connected case, it is obvious that each PE does a calculation on every time step. Therefore η=1.

Efficiency in the feed forward case is lower than the corresponding setup of network B. This is due to the fact that the size of the shift registers is determined by the size of the hardware. The nominal case yields high efficiency, though. During a single calculation, all the PEs work all the time, except for the output phase. During the output phase half of the PEs are idle. This yields

$$\eta = \frac{M}{M+1} .$$

As for networks A and B, efficiency in the back propagation case is low. On every stage of the back propagation process, one PE is calculating M results and then M PEs read an input. This happens M times for each layer, taking M+1 time steps per layer. This process is repeated K–1 times. This yields

$$\eta = \frac{M}{M+1} * \frac{1}{P} + \frac{1}{M+1} * \frac{M}{P} = \frac{2M}{(M+1) * P} \simeq \frac{2}{P} .$$

Calculating Complexity

Every PE in network C is connected to two bus segments. This yields a complexity measure of χ=2. Though network C needs $\sqrt{P}$ connections to the outside world, (Unlike networks A and B that need a constant number of connections), its internal wiring is not more complex than network B.

CONCLUSIONS

Summary of results

This paper has defined some measures for the evaluation of neural network hardware architectures. Those measures where used to analyze the three networks discussed, demonstrating their features and suggesting enhancements. Tables 1 to 3 summarize the results.

Table 1
Complexity summary - χ

Net A	Net B	Net C
1	2	2

Table 2
Convergence time summary - τ

Case	Net A	Net B	Net C
Fully Connected			
N=P	P	2P	N
N<P	P	P+N	N
N>P	$N * \left\lceil \frac{N}{P} \right\rceil$	$\left\lceil \frac{N}{P} \right\rceil (\left\lceil \frac{N}{P} \right\rceil + 1)P$	$N * \left\lceil \frac{N}{P} \right\rceil$
Feed Forward			
N=P	P	$\max_{i=1}^{K}(M_i)$	$2+\max_{i=0}^{K}(M_i)$
N<P	P	$\max_{i=1}^{K}(M_i)$	$2+\max_{i=0}^{K}(M_i)$
N>P	$P * \left\lceil \frac{N}{P} \right\rceil$	$\left\lceil \frac{\sum_{i=1}^{K} M_i}{P} \right\rceil \max_{i=1}^{K}(M_i)$	$\tau_{nom} \left\lceil \frac{K \max_{i=0}^{K}(M_i)}{P} \right\rceil$
Back Propagation			
N=P	$(K-1)P\max_{i=1}^{K}(M_i)$	$\sum_{i=2}^{K}(M_i(\frac{M_i}{2}+M_{i-1}))$	$K(\max_{i=1}^{K}(M_i))^2$
N<P	$(K-1)P\max_{i=1}^{K}(M_i)$	$\sum_{i=2}^{K}(M_i(\frac{M_i}{2}+M_{i-1}))$	$K(\max_{i=1}^{K}(M_i))^2$
N>P	$\tau_{nom} \left\lceil \frac{N}{P} \right\rceil$	$\sum_{i=2}^{K}(M_i(\frac{M_i}{2}+M_{i-1}))$	$K(\max_{i=1}^{K}(M_i))^2$

Table 3
Efficiency summary - η

In the next table: $C = \left\lceil \frac{N}{P} \right\rceil$; $M = \max_{i=1}^{K}(M_i)$

Case	Net A	Net B	Net C
Fully Connected			
N=P	1	$\frac{1}{2}$	1
N<P	$\frac{N}{P}$	$\frac{N}{P+N}$	1
N>P	$\frac{N}{C*P}$	$\frac{N}{P(C+1)}$	$\frac{N}{C*P}$
Feed Forward			
N=P	$\frac{N}{K \max_{i=0}^{K}(M_i)}$	$\frac{\sum_{i=1}^{K} M_{i-1}M_i}{P^2}$	$\frac{M}{M+1}$
N<P	$\frac{N}{P} * \frac{N}{K \max_{i=0}^{K}(M_i)}$	$\frac{\sum_{i=1}^{K} M_{i-1}M_i}{P^2}$	$\frac{M^2K}{(M+1)P}$
N>P	$\frac{N}{C*P} * \frac{N}{K \max_{i=0}^{K}(M_i)}$	$\frac{\sum_{i=1}^{K} M_{i-1}M_i}{C\,P^2}$	$\frac{M^2K}{C(M+1)P}$
Back Propagation			
N=P	$\frac{\sum_{i=2}^{K} 2M_iM_{i-1}}{P^2(K-1)M}$	$\frac{\sum_{i=2}^{K} 2M_{i-1}M_i}{P\sum_{i=2}^{K} M_i(0.5M_i+M_{i-1})}$	$\frac{2M}{(M+1)P}$
N<P	$\frac{\sum_{i=2}^{K} 2M_iM_{i-1}}{P^2(K-1)M}$	$\frac{\sum_{i=2}^{K} 2M_{i-1}M_i}{P\sum_{i=2}^{K} M_i(0.5M_i+M_{i-1})}$	$\frac{2M}{(M+1)P}$
N>P	$\frac{\sum_{i=2}^{K} 2M_iM_{i-1}}{C\,P^2(K-1)M}$	$\frac{\sum_{i=2}^{K} 2M_{i-1}M_i}{P\sum_{i=2}^{K} M_i(0.5M_i+M_{i-1})}$	$\frac{2M}{(M+1)P}$

Discussion

The speed measures that where established above, reveal that two among the networks are optimized for a specific neural network paradigm. Network A is optimal for fully connected architectures, while network B is suited for the layered cases. By considering problems that don't fit the hardware on a one to one mapping basis, it is shown that the one dimensional connectivity of networks A and B yield to the performance of network C.

The efficiency measure have demonstrated the claim that back propagation is hard to implement in hardware. The importance of flexibility in the interconnection is highlighted as well. Network C which is the most flexible, is more efficient than network A or B for all cases but one.

The complexity measure reveals that network A is more simple than networks B and C. Networks B and C have the same complexity, meaning that they are comparable on equal basis.

Beside using measures to predict performance, they can be used to suggest and asses improvements to the connectivity architecture. One such improvement regarding networks A and B is providing a "shortcut" communication channel that can be used to exclude surplus PEs, when the problem is smaller then the hardware. The improvements in performance can be seen by comparison to network C results.

Network C can benefit from separating the input and the output bus segments. This will enable to build up two communication channels per PE, rather than one. Again, the improvement can be predicted by comparing network C to network B.

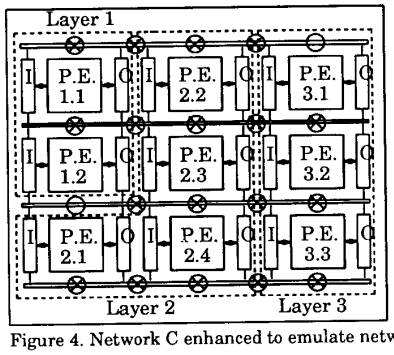

Figure 4. Network C enhanced to emulate network B.

Figure 4 shows the suggested modification to network C architecture. The soft-switches are set to emulate network B in an arrangement of three layered net.

REFERENCES

1 P. Treleaven, "Neurocomputers - Research notes", UCL, January 1989.

2 D. E. Rumelhart, J. L. McClelland et al. "Parallel Distributed Processing" Vol. 1, MIT press, 1986.

3 M. L. Minsky and S. A. Papert "Perceptrons", MIT press, 1988.

4 S. Jones, M. Thomaz, K. Sammut, "Linear systolic neural network engine", IFIP Workshop on Parallel Architectures on Silicon, Grenoble, France, December 1989.

5 D. J. Myers and G. E. Brebner, "Digital neural network on a SIMD processor", IFIP Workshop on Parallel Architectures on Silicon, Grenoble, France, December 1989.

6 J. Ouali and G. Saucier, "Customizable neural networks on silicon", IFIP Workshop on Parallel Architectures on Silicon, Grenoble, France, pp. 18-31, December 1989.

7 Y. Kashai and Y. Be'ery, "Comparing Digital Neural Network Architectures", IFIP Workshop on Silicon Architectures for Neural Networks, St. Paul-de-Vance, France, pp. II.3-II.24, November 1990.