Parallel Multi-Layer Neural Network Architecture with Improved Efficiency

David Hunter[1] and Bogdan Wilamowski[2]
[1]Brown Mackie College, [2]Auburn University
dshunter@brownmackie.edu, wilam@ieee.org

Abstract. **Neural network research over the past 3 decades has resulted in improved designs and more efficient training methods. In today's high-tech world, many complex non-linear systems described by dozens of differential equations are being replaced with powerful neural networks, making neural networks increasingly more important. However, all of the current designs, including the Multi-Layer Perceptron, the Bridged Multi-Layer Perceptron, and the Fully-Connected Cascade networks have a very large number of weights and connections, making them difficult to implement in hardware. The Parallel Multi-Layer Perceptron architecture introduced in this article yields the first neural network architecture that is practical to implement in hardware. This new architecture significantly reduces the number of connections and weights and eliminates the need for cross-layer connections. Results for this new architecture were tested on parity-N problems for values of N up to 17. Theoretical results show that this architecture yields valid results for all positive integer values of N.**

***Keywords*: BMLP, Cascade, Connected, FCC, Fully-Connected, Fully-Connected Cascade, MLP, Multi-Layer, Neural Network, Parity, PMLP.**

I. Introduction

NEURAL network architectures vary in complexity and efficiency. Over the past 3 decades, a significant amount of research has been invested in improving designs, training methods, and efficiency[1-20]. Research has thus far shown that some of the most efficient and reliable networks have fewer neurons, but also require multiple layers with connections between all layers [4-5, 10]. In fact, more and more complex non-linear systems, such as those in the oil industry, are described by a set of up to 25 differential equations. These complex systems are beginning to be replaced by neural networks [22-23]. Until now, neural networks that utilize connections across layers have been considered the most powerful [5, 10], however this type of design is especially difficult to implement in hardware [21]. A better solution is now available. This article will review the progression of network architectures from the simple Multi-Layer Perceptrop (MLP) to the Bridged Multi-Layer Perceptron (BMLP) to the Fully-Connected Cascade (FCC) Networks. This article will introduce a new architecture, the Parallel Muli-Layer Perceptron (PMLP) architecture, which utilizes linear neurons in parallel with non-linear neurons to eliminate cross-layer connections, resulting in an architecture that is just as powerful as previously discovered architectures, but is much more feasible to implement in hardware [21].

David Hunter: Brown Mackie College, 9050 W. Overland Rd., Suite 100, Boise, ID 83709 USA Phone: +1-208-994-2918
Bogdan Wilamowski: 200 Broun Hall, Auburn University, Auburn, AL. 36849-5201 USA. Phone: +1-334-844-1629.

II. Multi-Layer Perceptron with One Hidden Layer

For the MLP architecture with one hidden layer, a generalized solution for all parity-N cases is easily derived. First, the number of neurons in the network must be determined. For a MLP network with one hidden layer, the total number of neurons in the network, J, is determined by [8]:

$$J=N+1$$

where N is the parity number as well as the number of neurons in the hidden layer. The net values for the hidden neurons in a bipolar network are given by [8]:

$$net = (2i - N) + (2j - N - 1)$$

where:

$(2i - N)$ is the sum of inputs

$(2j - N - 1)$ is the bias

N is total number of inputs

i is the number of inputs with $+1$ signal

j is the hidden neuron number

The output neuron performs the AND operation on all neuron outputs from the hidden layer. The outputs of odd numbered neurons in the hidden layer are negated by the outputs of the even numbered neurons. This is accomplished by assigning a +1 weight to all inputs coming from the odd numbered neurons and a –1 weight to all inputs coming from the even numbered neurons if N is odd. These input weights are negated if N is even.

Figure 1 shows the bipolar implementation of a parity-8 network with one hidden layer. While this particular implementation is easy to train, it requires an excessive number of neurons for large parity-N or other complex problems as well as an excessive number of connections and weights. Although this architecture is functionally sound, it is not practical to implement in hardware , especially for large values of N [13].

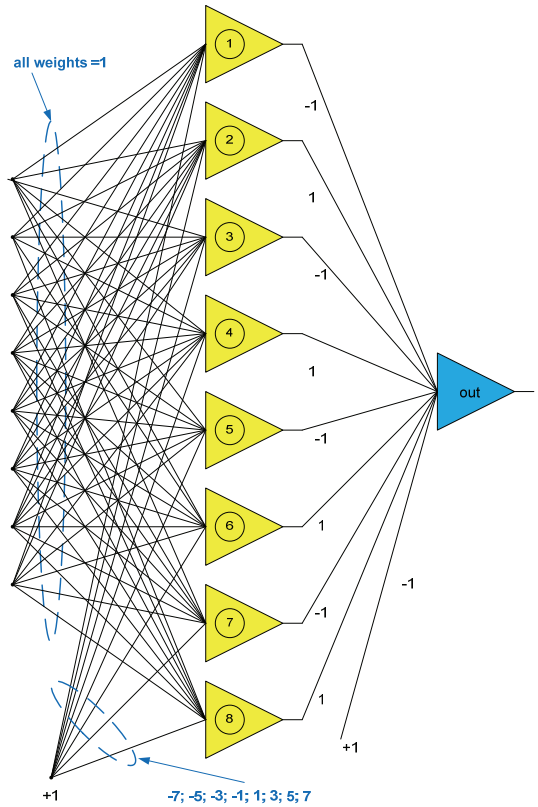

FIG. 1: MLP NEURAL NETWORK WITH ONE HIDDEN LAYER FOR THE PARITY-8 PROBLEM.

III. BRIDGED MULTI-LAYER PERCEPTRON WITH ONE HIDDEN LAYER

Use of a BMLP with one hidden layer can significantly reduce the number of neurons in the hidden layer as it allows connections between all layers [10]. In this configuration, all of the inputs are connected to the output neuron as well as the neurons in the hidden layer. The total number of neurons in the hidden layer of a BMLP parity-N network is:

$$n = \frac{N - \mathrm{mod}_2(N)}{2}$$

where N is the number of inputs and also the parity number. Notice that in this configuration, n only represents the number of neurons in the hidden layer, not the total number of neurons in the network. This more powerful network is able to solve the parity-(2^n-1) problem with only n neurons in one hidden layer. Addition of the output neuron brings the total number of neurons in the network to $n + 1$ [8].

Figure 2 shows the bipolar implementation of a BMLP parity-8 network with one hidden layer. Notice that this architecture reduces the required number of neurons by nearly half from the MLP architecture in Figure 1. While this reduction in neurons cuts the number of required connections and weights by over half, it does introduce N cross-layer connections where N is the parity number. These cross-layer connections do introduce some complexity in hardware implementation, making this architecture less than optimal and not the architecture of choice.

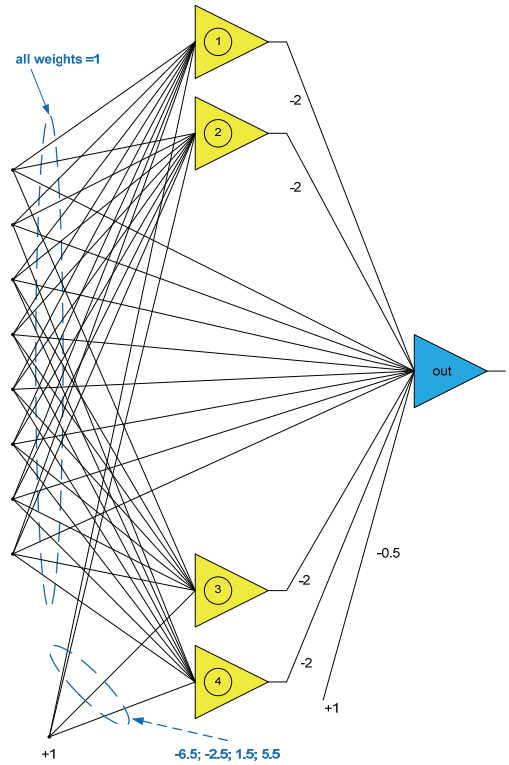

FIG. 2: FULLY CONNECTED BMLP BIPOLAR NEURAL NETWORK WITH ONE HIDDEN LAYER FOR THE PARITY-8 PROBLEM.

IV. BRIDGED MULTI-LAYER PERCEPTRON WITH MULTIPLE HIDDEN LAYERS

Use of a BMLP with multiple hidden layers further reduces the number of neurons required to solve a given parity-N problem [10]. In this configuration, all of the inputs are connected to the output neuron as well as the neurons in all hidden layers.

With two hidden layers, as seen in Figure 3, the largest parity-N problem that can be solved by this network is defined by:

$$N = 2(n_1+1)(n_2+1)-1$$

where n_1 and n_2 are the number of neurons in the first and second hidden layers, respectively. Wilamowski, Yu, and

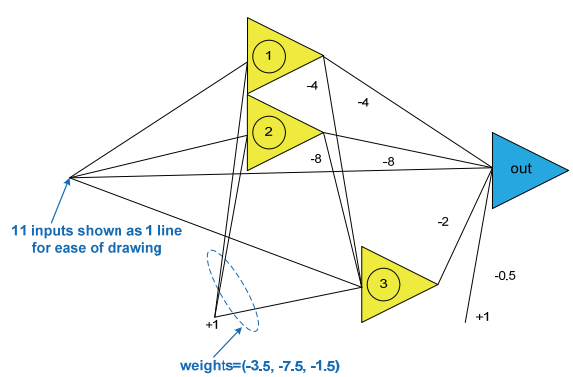

FIG. 3: FULLY CONNECTED BMLP NEURAL NETWORK WITH TWO HIDDEN LAYERS FOR THE PARITY-11 PROBLEM.

Chung show that this equation can be generalized for k

hidden layers. For k hidden layers with n_i neurons in each layer, the largest parity-N problem that can be solved is defined by [10]:

$$N = 2(n_1+1)(n_2+1)\ldots(n_{k-1}+1)(n_k+1)-1$$

where: N represents the parity number; n_1 is the number of neurons in the first hidden layer; n_2 is the number of neurons in the second hidden layer; and so on; until n_k which is the number of neurons in the last hidden layer.

By introducing additional hidden layers in the BMLP network, the power of the network is significantly increased. Larger parity-N problems can be solved with fewer neurons, but additional cross-layer connections are introduced. With each additional hidden layer, this architecture becomes more and more difficult to implement in hardware.

V. Fully Connected Cascade Network

Use of a fully connected cascade (FCC) network can further reduce the number of neurons in the network. This configuration has multiple hidden layers connected in cascade with only one neuron in each hidden layer. All of the inputs are connected to all neurons in the network. Additionally, the output of each hidden neuron is connected to the inputs of all neurons in front of it. By deduction, the total number of neurons in a FCC parity-N network is [8]:

$$J = \lfloor \log_2(N) \rfloor + 1$$

where N is the number of inputs and the parity number. Notice that the floor function, $\lfloor$(expression)$\rfloor$, is used in the previous equation. This function returns the largest integer less than or equal to the evaluated expression inside the brackets. Similar to the BMLP architecture, the largest parity-N problem that can be solved by this network is defined by:

$$N = 2^n - 1$$

where n is the total number of neurons in the network.

Figure 4 shows the bipolar implementation of a FCC network for the parity-8 problem. Notice that this network would also be capable of solving parity-15 by simply adding the additional inputs and changing the biasing weights.

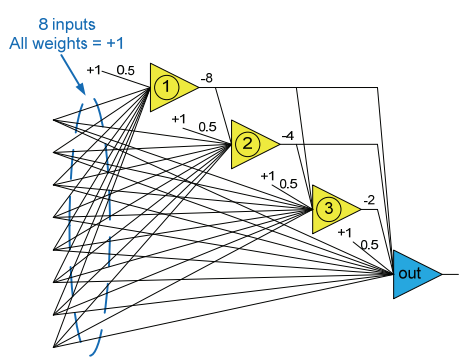

Fig. 4: Bipolar implementation of a FCC network for the parity-8 problem.

As previously discussed with the previous architectures, one of the difficulties in implementing the FCC network is the large number of total connections and weights and the large number of cross-layer connections. This large number of connections and weights make it difficult to implement this architecture in hardware.

VI. Simplified Fully Connected Cascade Network

With the introduction of a linear or summing neuron to the FCC network, the total number of connections and cross-layer connections is drastically reduced. This summing neuron is used as the single point for all input connections. This neuron performs the summing function for all inputs and passes the summed input to all forward neurons via a single connection.

Figure 5 shows the implementation of a linear neuron in the FCC network architecture to reduce needed connections.

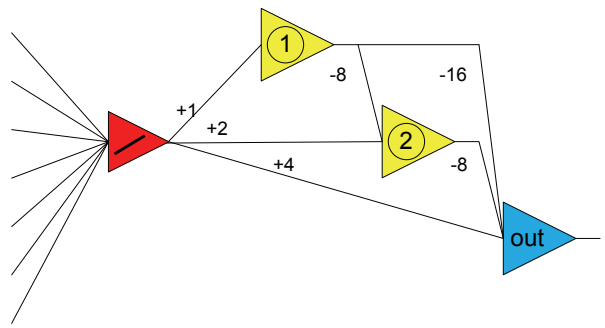

Fig. 5: Introducing the Linear Neuron into the FCC network for the parity-7 problem.

Notice that the number of connections and weights is reduced by (n-1)(N-1) where n is the number of non-linear neurons in the network and N is the parity number. This is a significant improvement over the previous network as its only practical limitation for hardware implementation is cross-layer connections.

VII. MLP, BMLP, and FCC Summarized

The three architectures shown so far represent a tremendous amount of research and are effective at solving the parity-N problem; however, each of these architectures presents a problem with practical implementation in hardware. Figure 6 compares architecture efficiency.

The MLP architecture is very inefficient and requires many neurons, typically N+1 to solve the parity-N problem. With this large amount of neurons comes a tremendous amount of connections and weights, making this architecture impractical to implement in hardware.

The BMLP architecture with one hidden layer improves upon the MLP by significantly reducing the number of required neurons, connections, and weights; however, it requires connections across hidden layers which are difficult to implement in hardware on a large scale. The BMLP with multiple hidden layers provides additional improvements over the BMLP with one hidden layer, but is still plagued by cross-layer connections.

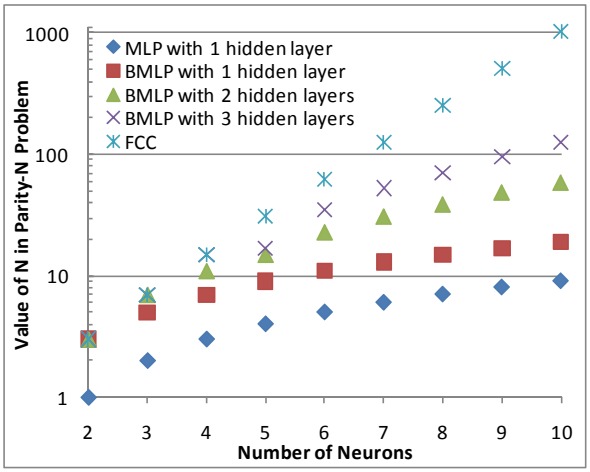

FIG. 6: EFFICIENCY COMPARISON OF VARIOUS NEURAL NETWORK ARCHITECTURES [7].

The FCC architecture provides a slight improvement in efficiency over the BMLP, requiring fewer neurons than the BMLP for the same parity-N problem. Yet, this architecture still requires connections across hidden layers as well as a large number of weights. With the implementation of a single linear neuron, used to sum all inputs and propagate the sum to all forward neurons, the number of cross-layer connections is drastically reduced. Of all the architectures presented thus far, this one is the most efficient and most feasible to implement in hardware.

Table 1 shows a summary of the number of neurons and weights required by different architectures [5].

Neuron/Weight ratios for different Parity Problems using neural network archtectures					
Architecture	**Parity-3**	**Parity-7**	**Parity-15**	**Parity-31**	**Parity-63**
MLP	**4/16**	**8/64**	**16/256**	**32/1024**	**64/4096**
BMLP-1 hidden layer	**3/14**	**5/44**	**9/152**	**17/560**	**33/2144**
BMLP-2 hidden layers	**N/A**	**3/27**	**5/88**	**7/239**	**11/739**
BMLP-3 hidden layers	**N/A**	**N/A**	**4/70**	**6/203**	**8/530**
FCC	**2/9**	**3/27**	**4/70**	**5/170**	**6/399**

TABLE 1: NEURONS AND WEIGHTS REQUIRED BY VARIOUS ARCHITECTURES.

VIII. THE PARALLEL MULTI-LAYER PERCEPTRON

While the BMLP and FCC architectures are very efficient in solving the parity-N problem, their major drawback is the number of weights required and the need for cross-layer connections. It is possible to get the benefits of these types of architectures with a significantly reduced number of weights and no cross-layer connections. This is possible in a new architecture which utilizes one linear neuron in parallel with 1 or more non-linear neurons per hidden layer. This architecture will be referred to as the Parallel Muli-Layer Perceptron (PMLP).

To understand the development of the PMLP, it is helpful to see its roots. Figure 7 shows the genesis of the PMLP in the FCC network. If one creates a network similar to that in Figure 5, for parity-15, the result can be seen in Figure 6 below. Note that the neuron numbers have changed to correlate with Figure 8 as this will be helpful in understanding the upcoming proof of concept.

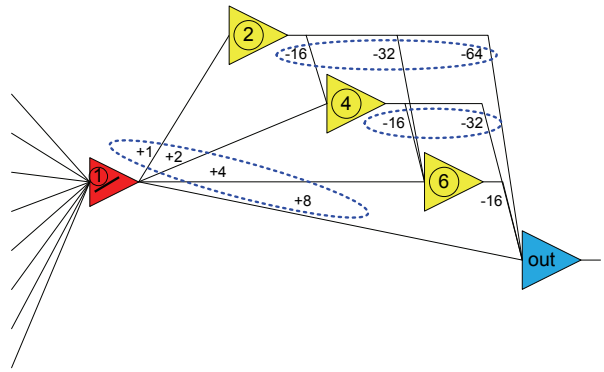

FIG. 7: MODIFIED FCC NETWORK WITH LINEAR NEURON FOR THE PARITY-15 PROBLEM.

If one examines the weights that are circled in Figure 7, one can see that neuron weights are multiplied by 2 as they cross each layer. For example the output of neuron 2 gets a weight of -16 going into neuron 4, a weight of -32 going into neuron 6, and a weight of -64 going into the output neuron. Likewise, the output of neuron 4 has a weight of -16 going into neuron 6 and a weight of -32 going into the output neuron. In other words, the initial weight, -16, is multiplied by 2 each time it crosses a hidden layer. Similarly, one may notice that the outputs of the linear neuron to all subsequent neurons are increased by a multiple of 2 for each subsequent layer starting with an initial weight of +1.

Understanding the pattern of weights in Figure 7 enables one to prove the concept of the new architecture which utilizes additional linear neurons and eliminates cross layer connections. This new architecture is called the Parallel Multi-Layer Perceptron. An example of a PMLP network for the parity-15 problem is seen in Figure 8.

Proof of concept for the PMLP is obtained by deriving the neuron input sum for neurons 2, 4, 6, and the output neuron for both Figure 7 and Figure 8. The derived inputs for each neuron can be found in Table 2 and Table 3.

$$\mathbf{Sum}_{N2} = (1) \times \mathrm{Sum}_{N1}$$
$$\mathbf{Sum}_{N4} = (2) \times \mathrm{Sum}_{N1} + (-16) \times \mathrm{Out}_{N2}$$
$$\mathbf{Sum}_{N6} = (4) \times \mathrm{Sum}_{N1} + (-32) \times \mathrm{Out}_{N2} + (-16) \times \mathrm{Out}_{N4}$$
$$\mathbf{Sum}_{OUT} = (8) \times \mathrm{Sum}_{N1} + (-64) \times \mathrm{Out}_{N2} + (-32) \times \mathrm{Out}_{N4} + (-16) \times \mathrm{Out}_{N6}$$

TABLE 2: DERIVED INPUTS FOR NETWORK IN FIG. 7

Notice that for the FCC network in Figure 7, it is relatively simple to derive the input sums for each neuron as the connections feeding each neuron are well defined and easily seen. In the derivations, a "Sum" term stands for the total summed input at a given neuron while an "Out" term stands for the bipolar output of the given neuron. Also, observe that if you look at the Sum equations in Table 2 that in each successive row the contribution (highlighted with blue dotted circle for clarity) from any given neuron is 2

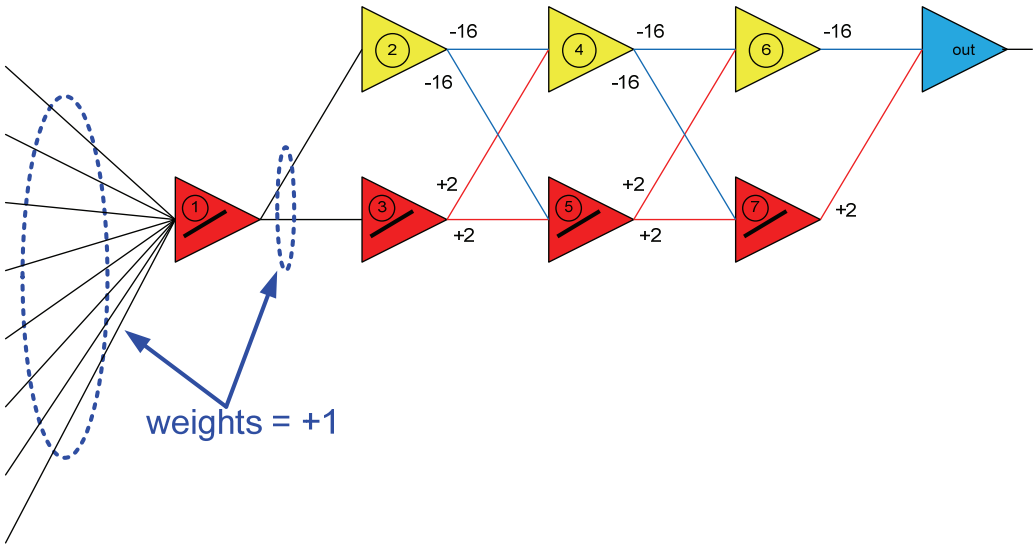

FIG. 8: BIPOLAR IMPLEMENTATION OF A PMLP NETWORK FOR THE PARITY-15 PROBLEM.

times that of the previous row. For example, the contribution of *N1* in Sum_{N4} is 2 x Sum_{N1}. For Sum_{N6}, it is 4 x Sum_{N1}. This pattern repeats and is key to determination of the weights for the linear neurons in Figure 8. Now, we will derive the input sums for the same neurons in Figure 8. If

$\mathbf{Sum_{N2}} = (1) \times Sum_{N1}$
$\mathbf{Sum_{N3}} = (1) \times Sum_{N1}$
$\mathbf{Sum_{N4}} = (-16) \times Out_{N2} + (2) \times Sum_{N3}$

Substitute $\mathbf{Sum_{N3}} = (1) \times Sum_{N1}$ into Sum_{N4} equation to obtain:

$\mathbf{Sum_{N4}} = (2) \times Sum_{N1} + (-16) \times Out_{N2}$
$\mathbf{Sum_{N5}} = (-16) \times Out_{N2} + (2) \times Sum_{N3}$

Substitute $\mathbf{Sum_{N3}} = (1) \times Sum_{N1}$ into Sum_{N5} equation to obtain:

$\mathbf{Sum_{N5}} = (2) \times Sum_{N1} + (-16) \times Out_{N2}$
$\mathbf{Sum_{N6}} = (-16) \times Out_{N4} + (2) \times Sum_{N5}$

Substitute $\mathbf{Sum_{N5}} = (2) \times Sum_{N1} + (-16) \times Out_{N2}$ into Sum_{N6} equation to obtain:

$\mathbf{Sum_{N6}} = (4) \times Sum_{N1} + (-32) \times Out_{N2} + (-16) \times Out_{N4}$
$\mathbf{Sum_{N7}} = (-16) \times Out_{N4} + (2) \times Sum_{N5}$

Substitute $\mathbf{Sum_{N5}} = (2) \times Sum_{N1} + (-16) \times Out_{N2}$ into Sum_{N7} equation to obtain:

$\mathbf{Sum_{N7}} = (4) \times Sum_{N1} + (-32) \times Out_{N2} + (-16) \times Out_{N4}$
$\mathbf{Sum_{OUT}} = (-16) \times Out_{N6} + (2) \times Sum_{N7}$

Substitute $\mathbf{Sum_{N7}} = (4) \times Sum_{N1} + (-32) \times Out_{N2} + (-16) \times Out_{N4}$ into Sum_{N6} equation to obtain:

$\mathbf{Sum_{OUT}} = (8) \times Sum_{N1} + (-64) \times Out_{N2} + (-32) \times Out_{N4} + (-16) \times Out_{N6}$

TABLE 3: DERIVED INPUTS FOR NETWORK IN FIG. 8

they are the same as those found in Table 2, then we have proven that the PMLP concept in Figure 8 yields identical results as the Modified FCC network in Figure 7 without the use of cross-layer connections.

Note that the four highlighted equations in Table 3 are identical to the four equations in Table 2. This proves that the networks in Figure 7 and Figure 8 produce the same output result for any given input. Thus the adaptation of the network in Figure 7 to that found in Figure 8 is complete.

The PMLP network found in Figure 8 provides many advantages over other network architectures. These advantages include:

- All inputs only have to connect to one neuron instead of many.
- All input weights are +1 and there are no biases on any of the neurons.
- The number of required weights is significantly reduced.
- There are no connections across hidden layers.
- Weights are uniform and fixed, making design simple.
- The only practical limitation to this design for implementation in hardware is the number of inputs into the first neuron.

Although the PMLP network uses almost twice the number of neurons as the FCC network for the same parity-N problem, half of the PMLP network's neurons are linear neurons which have a gain of +1 and are simple to implement. The benefit that these linear neurons add is that they eliminate the need for cross-layer connections. In essence, the linear neurons take the place of the cross-layer connections of the FCC network, significantly enhancing the practical implementation of this network architecture.

When working with this architecture, it is designed for Parity-(2^N-1) (i.e. Parity-15 for N=4 and Parity-31 for N=5). This design will require N linear neurons and N bipolar neurons. The first linear neuron stands alone in the first hidden layer and the last bipolar neuron stands alone in the output layer. All output weights for the linear neurons will be +2, except for the first linear neuron which is +1. All output weights for the bipolar neurons will be –(2^N). This architecture and design not only eliminates the need for cross-layer connections and significantly reduces the

number of total connections and weights, it is extremely easy to design for any positive integer, N. Results for this architecture were verified using the Neural Network Trainer [9].

Table 4 revisits the data summarized in Table 1 while adding PMLP data for comparison. As shown in Table 4, the PMLP architecture requires the same number of non-linear neurons as the FCC architecture, but requires significantly fewer connections and weights. Additionally, no cross-layer connections are required. Implementation of neural networks in hardware has been hindered by the difficulty of implementing large numbers of connections and weights in integrated circuits [13]. The PMLP architecture presented in this article provides solutions to these barriers. In addition to conquering these hardware barriers, this design significantly improves efficiency over the FCC architecture which has been the most efficient architecture to date.

Neuron/Weight ratios for different Parity Problems using neural network archtectures					
Architecture	**Parity-3**	**Parity-7**	**Parity-15**	**Parity-31**	**Parity-63**
MLP	**4/16**	**8/64**	**16/256**	**32/1024**	**64/4096**
BMLP	**3/14**	**5/44**	**9/152**	**17/560**	**33/2144**
BMLP-2 hidden layers	**N/A**	**3/27**	**5/88**	**7/239**	**11/739**
BMLP-3 hidden layers	**N/A**	**N/A**	**4/70**	**6/203**	**8/530**
FCC	**2/9**	**3/27**	**4/70**	**5/170**	**6/399**
PMLP*	**2/7**	**3/15**	**4/27**	**5/47**	**6/83**

*** Number of neurons for PMLP architecture does not include linear neurons.**

TABLE 4: NEURONS AND WEIGHTS REQUIRED BY VARIOUS ARCHITECTURES INCLUDING PMLP.

I. CONCLUSION

In conclusion, neural network architectures vary in complexity and efficiency. Decades of research has been invested in improving designs, training methods, and efficiency yielding designs such as the MLP, BMLP, and FCC. The most powerful and efficient networks have been those with fewer neurons and connections across layers. Hardware implementation of the previously mentioned architectures is not practical due to the large number of connections and weights. The PMLP architecture, which utilizes linear neurons in parallel with non-linear neurons, eliminates cross-layer connections and drastically reduces the total number of connections and weights, resulting in a more powerful, more efficient architecture which is the first architecture that is practical to implement in hardware.

REFERENCES

[1] D. Karras and I. Lagaris, ‘‘A Novel Neural Network Training Technique based on a Multi-Algorithm Constrained Optimization Strategy,’’ in IEEE, 1998, pp. 683-687.

[2] N. Gunaseeli and N. Karthikeyan, ‘‘A Constructive Approach of Modified Standard Backpropagation Algorithm with Optimum Initialization for Feedforward Neural Networks,’’ in Proc. International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, India, December 13-15, 2007, pp. 325-331.

[3] M. Di Martino, S. Fanelli, and M. Protasi, ‘‘Exploring and Comparing the Best ‘Direct Methods’ for the Efficient Training of MLP-Networks,’’ IEEE Transactions on Neural Networks., vol. 7, no. 6, pp. 1497–1502, 1996.

[4] S. Trenn, ‘‘Multilayer Perceptrions: Approximation Order and Necessary Number of Hidden Units,’’ IEEE Transactions on Neural Networks., vol. 19, no. 5, pp. 836–844, May 2008.

[5] B.M. Wilamowski, ‘‘Neural Network Architectures and Learning Algorithms,’’ IEEE Industrial Electronics Magazine, pp. 56–63, December 2009.

[6] H. Yu and B.M. Wilamowski, ‘‘Efficient and reliable training of neural networks,’’ in Proc. 2nd Conf. Human System Interaction, Catania, Italy, May 21–23, 2009, pp. 109–115.

[7] B. M. Wilamowski, " Challenges in Applications of Computational Intelligence in Industrial Electronics" ISIE10 - International Symposium on Industrial Electronics, Bari, Italy, July 4-7, 2010, pp. 15-22.

[8] B.M. Wilamowski, D. Hunter, and A. Malinowski, ‘‘Solving parity-n problems with feedforward neural network,’’ in Proc. IJCNN’03 Int. Joint Conf. Neural Networks, Portland, Oregon, USA, July 20–23, 2003, pp. 2546–2551.

[9] NNT-—Neural Network Trainer [Online]. Available: http://www.eng.auburn.edu/~wilambm/nnt/

[10] B.M. Wilamowski, H. Yu, and K. Chung, “Parity-N problems as a vehicle to compare efficiency of neural network architectures,” in The Industrial Electronics Handbook Vol. 5: Intelligent Systems, in print January 2011.

[11] B.M. Wilamowski, H. Yu, “Improved Computation for Levenberg Marquardt Training,” IEEE Trans. on Neural Networks, vol. 21, no. 6, pp. 930-937, June 2010.

[12] B.M. Wilamowski and H. Yu, “Neural Network Learning Without Backpropagation," IEEE Trans. on Neural Networks, vol. 21, no.11, Nov. 2010.

[13] B.M. Wilamowski, ‘‘Special neural network architectures for easy electronic implementations,’’ in Proc. Int. Conf. Power Engineering, Energy and Electrical Drives 2009, Lisbon, Portugal, Mar. 18–20, 2009, pp. 17–22.

[14] S. Ferrari, M. Jensenius, “A Constrained Optimization Approach to Preserving Prior Knowledge During Incremental Training,” IEEE Trans. On Neural Networks, Vol. 19, No. 6, pp. 996-1009, June 2008.

[15] C. Kim, J. Lee, “Training Two-Layered Feedforward Networks With Variable Projection Method,” IEEE Trans. On Neural Networks, Vol. 19, No. 2, pp. 371-375, February 2008.

[16] B.M. Wilamowski, N.J. Cotton, O. Kaynak, and G. Dundar, “Computing Gradient Vector and Jacobian Matrix in Arbitraryily Connected Neural Networks,” IEEE Trans. on Industrial Electronics, Vol. 55, No. 10, pp. 3784-3790, October 2008.

[17] N. Ampazis and S.J. Perantonis, “Two highly efficient second-order algorithms for training feedforward networks,” IEEE Trans. on Neural Networks, Vol. 13, No. 5, pp. 1064-1074, September 2002.

[18] A. Toledo, M. Pinzolas, J.J. Ibarrola, and G. Lera, “Improvement of the neighborhood based Levenerg-Marquardt algorithm by local adaptation of the learning coefficient,” IEEE Trans. on Neural Networks, Vol. 16, No. 4, pp. 988-992, July 2005.

[19] J.X. Peng, K. Li, and G.W. Irwin, “A new Jacobian matrix for optimal learning of single-layer neural networks,” IEEE Trans. on Neural Networks, Vol. 19, No. 1, pp. 119-129, January 2008.

[20] S. Wan and L.E. Banta, “Parameter incremental learning algorithm for neural networks,” IEEE Trans. on Neural Networks, Vol. 17, No. 6, pp. 1424-1438, November 2006.

[21] N.J. Cotton, B.M. Wilamowski, and G. Dundar, “A Neural Network Implementation on an Inexpensive Eight Bit Microcontroller,” 12th INES 2008 International Conference on Intelligent Engineering Systems, Miami, Florida, USA, February 25-29, 2008, pp. 109-114.

[22] B. M. Wilamowski and O. Kaynak, "Oil Well Diagnosis by sensing Terminal Characteristics of the Induction Motor," IEEE Transactions on Industrial Electronics, Vol 47, No 5, pp. 1100-1107, October 2000.

[23] B.K. Bose, “Neural network applications in power electronics and motor drives – An introduction and perspective,” IEEE Trans. on Industrial Electronics, Vol. 54, No. 1, pp. 14-33, February 2007.