Ambiguous Binary Representation in Multilayer Neural Networks

Cheng-Yuan Liou, Wen-Jen Yu
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan, 10764, R.O.C

ABSTRACT

We develop a side direction process to assist the back propagation learning algorithm to resolve the premature saturation problem. To build this side process, we explore the idea of unfaithful representation which has been introduced in the tiling algorithm[4]. The algorithm may grow to an unpredictably large network for a given pattern set. This unfaithful representation is equivalent to the ambiguous binary representation. Binary numbers are used to represent the output binary vectors of hidden layers. More training patterns of different classes map to the same binary number, more patterns are misclassified. Besides, the presence of ambiguous binary representations is also an important pointer of when and where we should add a new hidden neuron to the multilayer percetron. In this work, we explain the happening of ambiguous binary representation and develop a method to alleviate it. Using this method, both the number of ambiguous binary representations and the backpropagation learning time are drastically reduced.
Keyword : neural network, back propagation, tiling algorithm, premature saturation, unfaithful representation

1.Introduction to the ambiguous binary representation

The back propagation (BP) algorithm is widely used for finding optimum weights of multilayer neural networks in many pattern recognition subjects[6], [7].However, the critical problems of the algorithm are its slow learning speed and convergence to local minima[2], [3]. One of the major reason of these problems is "ambiguous binary representation". Binary numbers are used to represent the output binary vectors of hidden layers. More training patterns of different classes map to the same binary number, more patterns are misclassified. In this section, we illustrate the happening of ambiguous binary representation.

To solve the problem, we should release the factors that cause them. In [3],it is proposed that "premature saturation" is one of the reasons of these problems. Premature saturation is a phenomenon that the error of the neural network stays significantly high constant for certain unpredicted period of time during learning. It is also stated that premature saturation could be avoided by setting the proper maximum value of initial weights. From our later discussion, we recommend to use the hyperplane grid or random bundaries for each layer as the initial weights to bound and discriminate all patterns as uniformly as possible. In our study premature saturation is the result of bad encoding of the pattern space. For example, the first hidden layer encodes the input pattern space to binary numbers (or binary vectors) of length m , where m is the number of neurons in the first hidden layer. Bad encoding causes the patterns of different classes map to the same binary number. We call this kind of binary numbers "ambiguous binary representations". Ambiguous binary representations do result in the premature saturation. Note that this encoding also preserves the input pattern neighbors in the neighborhood of Hamming tree for each layer.

One approach to the problem is using constructive algorithm : it begins with the smallest possible network and gradually increases the size if the error is kept above a level. The presence of ambiguous binary numbers is also an important pointer of when and where we should add a new hidden neuron to the MLP. By analyzing each individual ambiguous binary representation, we can constructively initialize the weights connected with the newly added neuron.

In the following we present simple cases to clearly illustrate the ambiguous binary representation problem. Consider a classification problem consisting in assigning training patterns of R^2 to 2 predetermined classes, i.e., class 0 and class 1. The training patterns of this classification problem are shown in Fig. 1. In Fig. 1(a), "x" represents patterns that belong to class 1, "o" represents patterns that belong to class 0. Assume that we already implement this classification using a three-layer neural network with 2 input units, 3 neurons in the first hidden layer, 3 neurons in the second hidden layer, 1 output neuron, and 3 bias units, as illustrated in Fig. 1(b). In this figure L_1, L_2, L_3 represent the first, second, third neuron in the first hidden layer
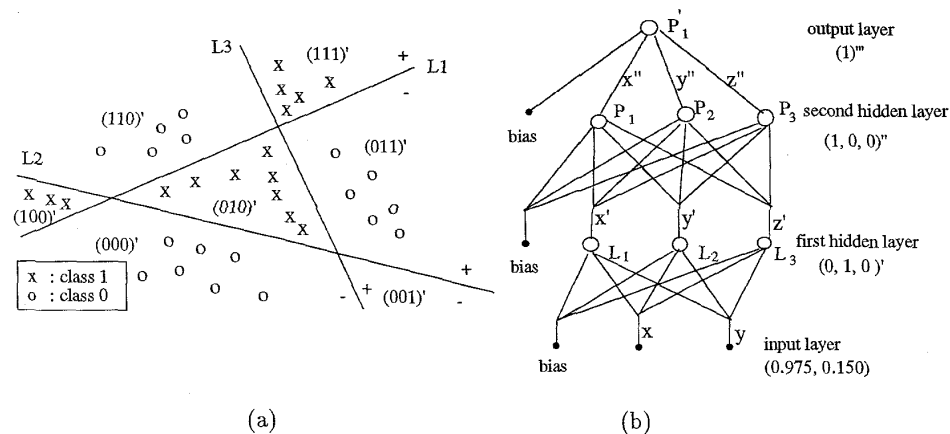
Fig. 1. The three-layer neural network and the discriminated regions of the first hidden layer.

and P_1, P_2, P_3 represent the first, second, third neuron in the second hidden layer and P_1' is the output neuron.

Let the activation function used in this multi-layer neural network be the hard-limited function. Then the function of every neuron is $F(X) = \{0$ if $W \cdot X + w_0 < 0, 1$ if $W \cdot X + w_0 > 0\}$, where X is the input vector, W is the weight vector of the hidden neuron or output neuron, and w_0 is the bias of the hidden neuron.

Now let's state the detailed function of the first hidden layer. Assume that Fig. 1 shows a perfect trained result of the BP learning algorithm. The three lines (or boundaries) shown in Fig. 1 are the lines corresponding to the three hidden neurons in the first hidden layer. The positive $(F(X) > 0)$ and negative sides $(F(X) < 0)$ of each line are also displayed. The function lines of the first, second, and third neuron of the first hidden layer are $\{L_1 : -1.3 * x + 3 * y = 3.9, L_2 : x + 4 * y = -3, L_3 : 2.3 * x + y = 1.3\}$, where x and y are the inputs to the input layer. The coefficients of x and y are the trained weights of the neurons in the first hidden layer.

The center region (polyhedral) in Fig. 1 lies on the negative part of L_1, positive part of L_2, and negative part of L_3. Thus this region is represented by the output binary vector $(010)'$ of the first hidden layer. All of the seven binary representation regions discriminated by the first hidden layer shown in Fig. 1 are the finest *elementary building blocks* for all upper layers. All upper layers use the various combinations of these finest building blocks to achieve the classification performance. This means, when there exists a mixed class region in any of these seven elementary regions, such as the elementary region $(011)'$ contains both types of patterns "x" and "o", the misclassification will persist with this ambiguous elementary region. This will cause difficulty in including it for all upper layers no matter

how we tune the weights between the first hidden layer and the output layer.

Notice that input neighborhood patterns map to Hamming neighbors. For example, the binary vector for the center region is $(010)'$ and the binary vector for the right center region is $(011)'$ as in Fig.1. These two connected neighborhood regions have a common boundary L_3 and have a Hamming distance of only one bit. All connected neighborhood regions have an exactly one bit Hamming neighborhood distance. Far and unconnected regions may have maximum Hamming distance of all three bits. By the first hidden layer, the 2-class pattern topology in input space is easily converted to the Hamming topology in unit hypercube. In Hamming space, the neighborhood is counted in term of Hamming distance. Since these regions are finite and countable for the MLP, the hyperspace Hamming neighbors can be displayed by a Hamming tree structure with successive fans [6]. With this trained MLP, all output vectors of the first hidden layer can be presented as binary numbers of length of 3 (3 is the number of neurons in the first hidden layer). The total capacity of these binary representations is $2^3 = 8$. Note that there are seven representation regions can be allocated in this plane, see [5] for details. In most cases the number of represenation region is much less than the number of the training patterns when these patterns are cluster type. Thus monitoring the Hamming tree during BP learning is feasible in our processing.

Every training pattern map to one corner point of the cube as shown in Fig. 2. In general, the function H_1 of the first hidden layer is a mapping $H_1 : R^2 \rightarrow \{0,1\}^3$, where 2 is the dimension of the input pattern space and 3 is the number of neurons in the first hidden layer.

Now let's state the function of the second hidden layer. The function H_2 of the second hidden layers is $H_2 : \{0,1\}^3 \rightarrow \{0,1\}^3$. That is, every neuron in
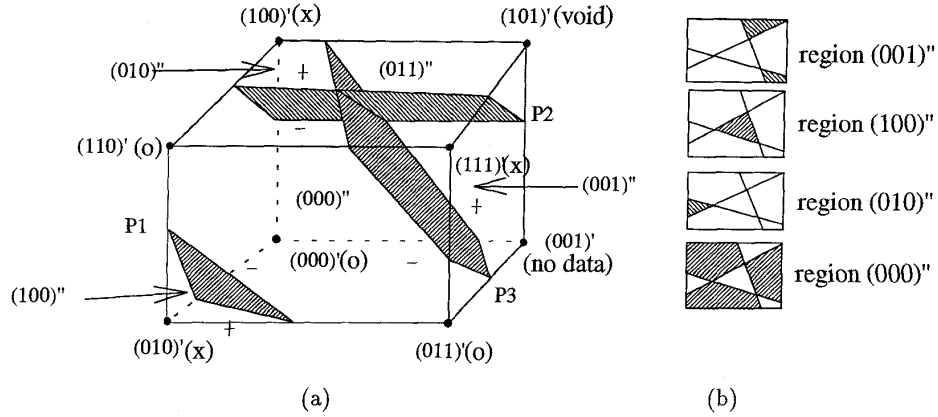
380

Fig. 2. The discriminated regions of the second hidden layer.

the second hidden layer decides a plane to divide the unit cube of the first hidden layer. The plane (or boundary), P_1, corresponding to the first neuron in the second hidden layer is shown in Fig. 2. This neuron's output for $(010)'$ is "1", all others are "0". Assume that the three planes shown in Fig.2 corresponding to the three neurons in the second hidden layer are P_1, P_2, and P_3. The plane functions of P_1, P_2 and P_3 are $\{P_1 : -0.9 * x' + 1.5 * y' - z' = 0.95, P_2 : 2.5 * x' - 3.7 * y' + 0.37 * z' = 3.52, P_3 : 2.5 * x' - 2 * y' + 4 * z' = 3.1\}$, where x' is the output value of the first neuron L_1, y' is the output value of the second neuron L_2, and z' is the output value of the third neuron L_3 in the first hidden layer. The coefficients of x',y', and z' are the trained weights. Then the training patterns which map to the binary number $(010)'$ in the first hidden layer will cause the first neuron in the second hidden layer "ON". The training patterns which map to the binary vector $(111)'$,$(001)'$, and $(101)'$ of the first hidden layer will cause the third neuron in the second hidden layer "ON".

Fig. 2 shows that the cube is divided into five regions by the three planes, P_1, P_2, and P_3. Each region is represented by an output binary vector of the second hidden layer. This means that each region is a combination of several finest element regions of the first hidden layer. The region $(001)''$ includes vertexes $(111)'$ and $(001)'$ which are two element regions of the first hidden layer. The region $(011)''$ includes $(101)'$. The region $(010)''$ includes $(100)'$. We can see that the four regions in Fig.2(b) which are discriminated by the second hidden layer are elementary building blocks of the output layers. This means that any ambiguous binary representation happened in this layer will cause misclassification in the output layer.

The outputs of the second hidden layer form a unit hypercube, too. The output neuron also decides a plane to divide this cube. This is shown in Fig. 3 and the plane function of the output neuron

is $\{P_1' : 5 * x'' + 6 * y'' + 6 * z'' = 3\}$, where x'' is the output value of the first neuron P_1, y'' is the output value of the second neuron P_2, and z'' is the output value of the third neuron P_3 in the second hidden layer. The coefficients of x'', y'', and z'' are the trained weights.

In each hidden layer, every binary number represents how many training patterns of class 0 map to it and how many training patterns of class 1 map to it. If there are only one kind of pattern of class 0 (or class 1) mapto it, we say that this binary number's type is 0(or 1) and homogenous. Allbinary representations in Fig. 1(a) are homogenous representations for thefirst hidden layer. If there are both patterns of class 0 and class 1 map to asame binary number, we say that its type is ambiguous or unfaithful[4]. As anexample, the center region in Fig. 4(a) is in this situation, i.e., the binarynumber $(010)'$ of the first hidden layer is ambiguous. The ambiguousbinary number means that there are patterns belong to different classes mapto the same output binary vector of the first hidden layer or any otherhidden layer. If the region $(010)'$ of the first hidden layer maps tooutput $(0)'''$, the patterns of class 1 in this region will bemisclassified. If this region maps to output $(1)'''$, the patterns of class 0 in thisregion are misclassified. This misclassification can not be corrected orlearned by training the all weights between the output layer and the firsthidden layer. The region $(001)''$ shown in Fig. 4(b) is also an ambiguousregion for the second hidden layer even when all regions of the first hiddenlayer are homogeneous. Fig.4(c) shows a compound case.

The four elementary regions discriminated by the second hidden layer are showing in the Fig. 2. All of the 4 binary representations are elementary building blocks of the upper output layer. In our case this means the output layer P' can discriminate patterns based on combinations of these 4 elementary regions. For example, when there are mixed patterns in theregion $(001)''$ shown in Fig. 4(b),
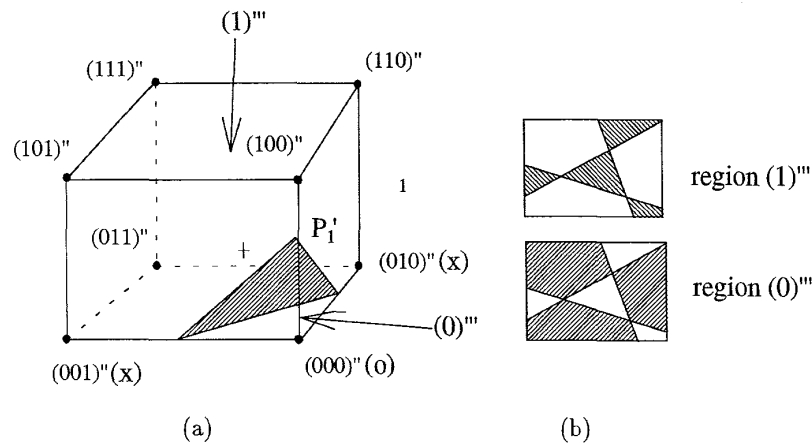
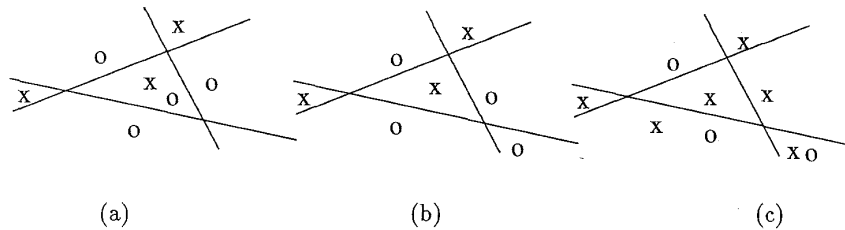Fig. 3. The discriminated regions of the output neuron.

Fig. 4. Three types of different ambiguous representation.

the BP can not converge by training the weights between the second hidden layer and the output layer.

The fact is that the patterns which cannot be discriminated in the first hidden layer will mislead all upper layers' weights. This is because the binary represented regions of the lower layer are elementary building blocks of the regions of all its upper layers. This fact provides us the clue to improve the performance of the BP learning algorithm.

By monitoring the ambiguous binary representations for each hidden layer, we always can update the weights of the lowest hidden layer which contains any ambiguous region to improve the performance without tuning the weights of all homogeneous layers below it. The learning starts from the most important first hidden layer which provides the finest elementary regions until all binary vectors of that layer represent homogeneous regions. Then we train the weights of the next upper layer and frozen all lower homogeneous layers' weights. An extra neuron is added to a hidden layer only when it fails to be homogeneous during the learning.

2. The argumented learning algorithm and simulation

From Fig. 1, 2, and 3, each upper layer uses the successively combined discrete regions of its lower hidden layers to achieve the performance. The am-

biguous regions in the lower layers can never be improved by adjusting the upper layers' weights. The ambiguous regions of the lower layers will confuse and mislead all its upper layers. According to the idea stated in the last section, to solve the ambiguous binary representation in the first hidden layer is more important than to solve the ambiguous binary representation in the second hidden layer. Thus we can solve the ambiguous binary representation problem by starting solving the ambiguous binary representation from the lowest first hidden layer.

Now let's consider how to solve the ambiguous binary representation in the first hidden layer. Initially we may train a multi-layer neural network by the formal BP learning algorithm[6], [7]. When no further improvement is seen in the level of error, we monitor the ambiguous binary representations of the first hidden layer (we check the presence of the ambiguous binary representations of the first hidden layer). If the error is small enough, we stop. Otherwise, only the lowest layer with ambiguous representations will be trained using the BP algorithm. If there exists stubborn ambiguous binary representations we add new neurons to the first hidden layer in an attempt to alleviate the number of ambiguous binary representations and include more pattern informations flowing to the upper layers. To tackle this stubborn ambiguous binary representation problem, we first identify the regions and patterns belong to this ambiguous binary numbers
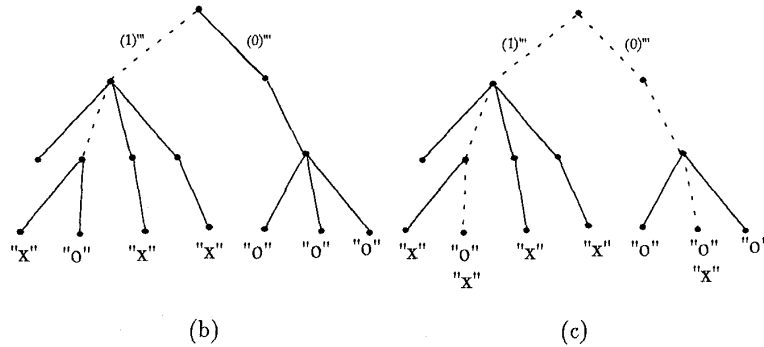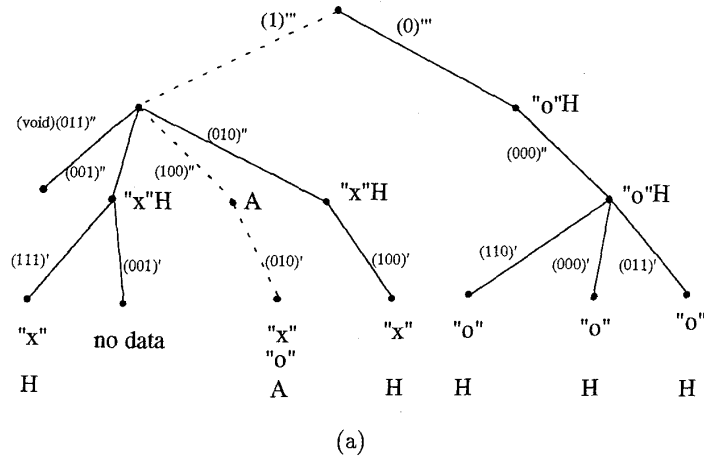
Fig. 5. The Hamming trees correspond to the cases in Fig. 4.

by calibrating the input pattern data. We then initialize the weights of the newly added neurons. This initialization is focused on adding newdiscriminative boundaries through each individual ambiguous region and keepingall other trained hyper boundaries. Each ambiguous region can be analyzedindividually. There are many methods for resolving different kinds ofambiguous regions[4] .We can also apply the method in [1] to any individual ambiguous region.

The learning starts from the first hidden layer and continues to the next upper layer until there exists no such ambiguous representation in any layer. Monitoring the ambiguous binary representation for each layer to help us understanding and improving the progress of the learning, is the introduced side process in this argumented learning algorithm. This will cost extra computations and memories. When the patterns are well cluster type, the memory size for use in memorizing the ambiguous representations is much less than the size of the total patterns. A Hamming tree can point the presence of ambiguous binary representations during the formal BP learning[6]. This tree is simple and clear for our purpose. The extra computations for obtaining this tree are on the order of the patterns with a

well sorting algorithm. Fig. 5 shows the Hamming trees for several kinds of ambiguous regions as in Fig. 4. The ambiguous regions of the lower layers will propagate upwards along the tree to all linked regions of the upper layers. The presence of an ambiguous region in a layer may come from its own wrong combination of the homogeneous regions of the lower layer as in Fig. 4(b) and 5(b). It also may come from carrying on an ambiguous region of the lower layer as in Fig. 4(a) and 5(a). For convenience, we may train a network by the formal BP algorithm and monitoring the occurrence of premature saturation (The error of the network is constantly high for a period of epoch). Once we detect its occurrence we trace the presence of the ambiguous binary representations and correct them as possible from the lowest ambiguous layer. The correction may be done by applying the formal BP to that layer and all its upper layers. A new neuron is added only when a stubborn ambiguous representation can not be resolved by the BP algorithm. The maximum number of hidden neurons follows the discussion in [5].

Fig. 6 shows the simulation result for a two-spirals problem. We can see that the argumented method is significantly better than the formal BP
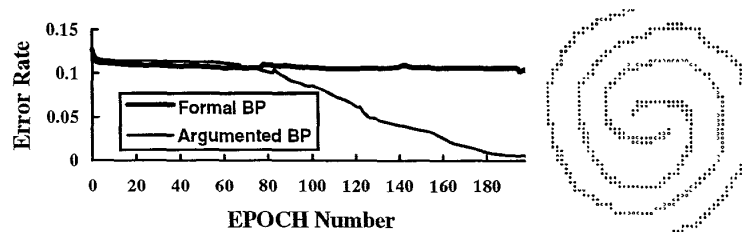
383

Fig. 6. The simulation result for this two-spirals' problem.

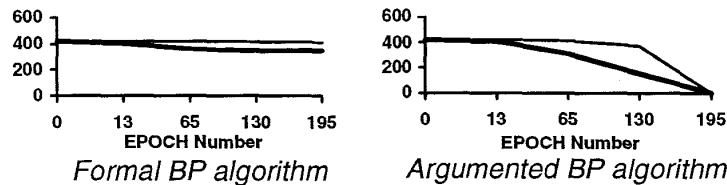Formal BP algorithm   Argumented BP algorithm

Fig. 7. Total amount of patterns belongs to the ambiguous regions in each layer. The thin line is for the second hidden layer, the thick line is for the first hidden layer.

algorithm. The problem has 419(209+210) patterns which form the two spirals. The network has two hidden layers with 5 neurons in the first hidden layer and 25 neurons in the second hidden layer. Fig. 7 shows the total amount of patterns belong to the ambiguous regions in each hidden layer during training. Using the formal BP algorithm and the argumented BP algorithm respectively. Both trainings start with small random weights.

3. Discussions and conclusions

In this work, the ambiguous binary representation problem in the multilayer perceptron is discussed. It is one of the major reasons for the learning problem of the formal BP method. We propose an argumented method to alleviate the ambiguous binary representation. Simulation results show that this argumented method can drastically reduce the learning time for many kinds of extremely difficult classification problems.

In many other works people initialize multi-layer neural networks with small random weights. According to the idea that the finest binary regions of the first hidden layer which provide very discriminative pattern informations to all the upper layers and ease the learning for those layers, we would suggest to initialize the weights with regular or random discriminative boundaries which evenly spread over the pattern ranges in each layer. For each second or higher hidden layer, these boundaries would divide the unit hypercube evenly. These boundaries will meet the suggested initialization method in[3] . Using small random weights as initial weights, the ambiguous binary representation problem is much severe. This is because most discriminative regions

may concentrate near the origin in the beginning of the learning and they progress slowly to cover the wide ranges of patterns. This will cause insufficient pattern informations flowing to the upper layers to afford the beginning learning informal BP learning.

Since the formal BP learning algorithm uses the error of the whole network as its all updation information, any local ambiguous region cannot be accurately corrected by using this error only. The BP algorithm will even revise all layers when lowest layers are all homogeneous. This makes no sense. Since we can locate ambiguous regions accurately, we can save computations, speed the learning, improve the performance in the argumented method.

References

[1] T. Denoeux, and R.Lengell, "Initializing back propagation networks with prototypes," Neural network, vol. 6,pp. 351-363, 1993.

[2] D. R. Hush, and B. G. Horne, "Progress in supervised neural networks," IEEE Signal Processing mag., pp. 8-39, 1993.

[3] Y. Lee, S. H. Oh, and M. W. Kim, "An analysis of premature saturation in back propagation learning," Neural Network, vol. 6, pp. 719-728, 1993.

[4] M. Mezard and J. P. Nadal, "Learning in feedforward layered networks: the tiling algorithm," J. Phys. A: Math.Gen. vol. 22, pp. 2191-2203, 1989.

[5] G. Mirchandeni and Wei Cao, "On hidden nodes for neural nets," IEEE Trans. on Circuits and Systems, vol.36, no. 5, pp.661-66198

[6] T. J. Sejnowski, and C. R. Rosenberg, "NETtalk:a parallel network that learns to read aloud," The Johns Hopkins University Electrical Engineering and Computer Science Tech. Report, JHU/EECS-86/01, 1986.

[7] B. Widrow, and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," IEEE Trans. on Computer, pp. 25-39, March, 198.