A NEURAL NETWORK ARCHITECTURE TO IDENTIFY THE BONE TISSUE FOR SOLID FREEFORM FABRICATION

Nena Marin
Richard H. Crawford
Department of Mechanical Engineering
The University of Texas at Austin

ABSTRACT

Computed tomography produces sets of tomograms for medical interpretation. Typical interpretation consists of imaging and simple observation on a 2-D display screen, so that feature extraction and tissue differentiation is based primarily on human expertise. Solid freeform fabrication offers the promise of fabrication of prostheses based on actual patient anatomy. Use of CT data for this purpose requires automated interpretation.

This paper presents a system architecture based on neural networks for the segmentation and classification of tissues of interest in tomograms. This approach produces a quantitative recovery of the available information by applying a feed-forward neural net trained with the back-propagation algorithm. The neural network architecture selected was tested on fabricated CT image matrices of the lower extremity.

Index terms: Human anatomy, modeling; Prosthesis, manufacturing; Prosthesis, rapid prototyping; Tissue identification, neural networks

INTRODUCTION

Solid Freeform Fabrication technologies offer several options for direct manufacturing of tooling and functional prototypes from geometric data. One option of interest in the area of production of medical prosthesis and implants is selective laser sintering (SLS). The ability of the final product to accurately represent the replaced human part depends theoretically on two factors: an accurate source of data and the process to produce the prosthesis from such data. Magnetic resonance (MR), positron emission tomography (PET) and computed tomography (CT) are traditional sources of data which produce a set of tomograms representing parallel sections of the body. Each cross-section image or slice is a 2D image matrix of quantified gray scale values. Generally, this information is used to segment and reconstruct a 3D graphics image of the anatomy in question. This task has proven difficult for conventional processing techniques. Levy et al. (1994) found "the utility of 3D CT reconstructions in head and neck imaging controversial". Our purpose in this research is to present a system architecture based on neural networks for fully automated segmentation and recognition of tissues of interest in CT cross-section images.

Computed Tomography

CT is one of the most widely used medical imaging techniques. CT, like conventional tomography, images a section or slice of the patient. This is accomplished by obtaining a series of different angular projections, or views of the section, and reconstructing a two-dimensional image of the slice from this series of one-dimensional projections. However, with CT, in contrast to conventional tomography, X-rays do not pass through neighboring anatomy, only through the section of interest. Thus, the reconstructed image does not suffer from the superimposition problem and is better able to demonstrate slice anatomy.

The reconstructed image is a two-dimensional array of quantified gray-scale values or picture element (pixels). These pixel values are directly related to the linear-attenuation coefficients of the corresponding values of the slice:

$$p = 1000\left(\frac{v}{v_{water}} - 1\right) \tag{1}$$

where p is the pixel value in Hounsfield units, v is the average linear-attenuation coefficient of the volume element (voxel) represented by the pixel, and v_{water} is the linear-attenuation coefficient of water, where both are evaluated for the effective energy of the beam exiting the patient. Thus, water has a CT number of zero, and a region with a CT number of 100 Hounsfield units has a linear-attenuation coefficient that is 10% greater than the linear-attenuation coefficient of water.

Each voxel is a right square prism. The height of the voxel is determined by the slice width and the square base by the pixel size. The pixel dimension, s_p, is in turn related to the display field of view, FOV, and the image matrix size, s_m:

$$s_p = \frac{FOV}{s_m} \tag{2}$$

The scan FOV is the region or circle over which data are acquired during a scan. On most scanners, the slice width and the FOV can be selected. This allows one to focus the reconstruction on the anatomy of interest.

CT images typically have 12 bits per pixel that represent numbers from −1000 to 3095, a total of 2^{12} or 4096 different gray-scale values. Air typically has a CT number of −1000, whereas fat is in the range of −80 to −100, soft-tissue structures 10 to 80, and bone 400 to 3000. The number for a given type of tissue will vary from scanner manufacturer to manufacturer and for a given scanner is highly dependent on calibration. By concentrating on a single CT scanner, namely the GE 9800^{TM}, this research combines the use of this CT number or gray-scale value and the learning capabilities of artificial neural networks to automate the medical interpretation expertise for identification of bone tissue. The segmented tissue can then be used to build the computer graphics 3D image data to drive the SLS process.

Artificial Neural Networks

Artificial neural networks, or neural nets, simulate the functioning of the brain directly on a computer. Neural nets are typically composed of interconnected units, which serve as model neurons. The function of the synapse is modeled by a modifiable weight, which is associated with each connection. Most artificial networks do not reflect the detailed geometry of dendrites and axons, and they express the electrical output of a neuron as a single number that represents the rate of firing-its activity (Hinton, 1992). Each unit converts the pattern of incoming activities that it receives into a single outgoing activity that it broadcasts to other units. The unit performs this conversion in two stages. First it multiplies each incoming activity by the weight on the connection and adds together all these weighted inputs to get a quantity called the total input. Second, a unit uses an input-output function that transforms the total input into the outgoing activity. The behavior of an artificial neural network depends on both the weights and the input-output function

^{TM}GE 9800 CT scanner is a product property of General Electric, Milwaukee, WI.

that is specified for the units. This function typically falls into one of three categories: linear, threshold or sigmoid. For linear units, the output activity is proportional to the total weighted input. For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For sigmoid, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

The perceptron was one of the earliest neural network models. The amazing property of perceptron learning is that whatever the perceptron can compute, it can learn to compute. Single-layer perceptrons can only build linear separating hyperplanes, and thus can separate only linearly separable classes. This limitation, pointed out by Minsky and Papert (1969) at the end of the 1960's, led to a substantial decrease in interest in neural network research.

A way to overcome this limitation was proposed by Werbos and formulated as the *backpropagation learning rule* of Rumelhart, Hinton, and Williams (Hinton, 1992). The backpropagation model is based on two principles:

- To overcome the limitations of the single-layer perceptron, all that is necessary is to insert one or more additional layers between input and output. These layers consist of processing units with nonlinear activation functions, typically sigmoid functions. Arbitrary convex classes can be separated by a network with one such hidden layer, and arbitrary nonconvex classes by a network with two hidden layers.
- The delta rule can be used to learn output layer weights. The remaining weights can then be learned by recursive application of the chain rule for computing derivatives.

For a two-layer network, the rule is the following:

$$\begin{aligned}\frac{\partial E}{\partial w1_{ji}} &= \sum_k \sum_j \frac{\partial E}{\partial o_k} \frac{\partial h_j}{\partial \sum_i w1_{ji}x_i} \frac{\partial \sum_i w1_{ji}x_i}{\partial w1_{ji}} \\ &= \sum_k \sum_j (o_k - y_k) w2_{kj} h_j (1 - h_j) x_i \end{aligned} \qquad (3)$$

where E represents the error, the derivative of E is the difference between the actual and the desired activity, x_i are the input unit activations, h_j are the hidden-unit activations, o_k are the output unit activations, $w2_{kj}$ are the output layer weights, and $w1_{ji}$ are the hidden-layer weights. These simple principles have constituted a very successful model that is the basis of a large number of applications.

Project Description and Scope

As a first step towards rapid prototyping of prostheses, we have investigates the potential utility of neural networks in the classification of tissue (more specifically, bone) from CT images. A CT slice or tomogram is a 2D digital image composed of a matrix of x,y pixel points and a gray-scale value or CT number. Each tissue type is known to have a different range of gray scale values. But the range is greatly dependent on calibration of the CT scanner, etc. This project goal is to use this CT number as a first determinant in classifying tissue. The 2-D CT image matrix is fed into a neural network designed to classify tissue types based on cases presented during the learning stage. The advantage is then that the neural network will learn to classify based on actual cases and build its own thresholds based on that information. Just as X-ray technicians and medical professionals have learned to identify and interpret images from training and experience, so will the neural network. A neural network offers the ability to learn to classify the inputs it is trained on, and to generalize and classify inputs that it has not yet seen. The ideal situation is to have the computer take in a series of parallel slices of the body or a section of the body and be able to obtain a 3-D view of that section where the computer has identified trouble areas. Currently, mathematical models of the image and its features require long complicated algorithms to accomplish this task. Human expertise is also required to provide thresholds for patients in different age and health categories.

For this project the input training image matrices as well as the testing image matrices were fabricated. The project focused on identifying bone tissue. The neural network used is a 3 layer feed-forward network trained using the Back-Propagation Method. Since the output varies continuously but not linearly as the input changes, the activation function used is a sigmoid. The delta rule is used to calculate errors and propagate them back through each layer at each iteration.

SYSTEM ARCHITECTURE

Because tomography data is proprietary, assumptions were made based on the technical specifications available about tomograms. The image matrix size is assumed to be 512x512. As described in the previous section, a tomogram is actually a 3-D slice whose size is determined by the pixel size and the slice width. For the purposes of this project a single slice will be treated as a 2-D image matrix. Figure 1 shows typical tomograms at levels K and L of the lower leg region. Tomogram K shows the fibula and tibia tissue cross-sections. The corresponding volumetric data consists of a matrix of 2D pixel locations and a CT or gray scale value.

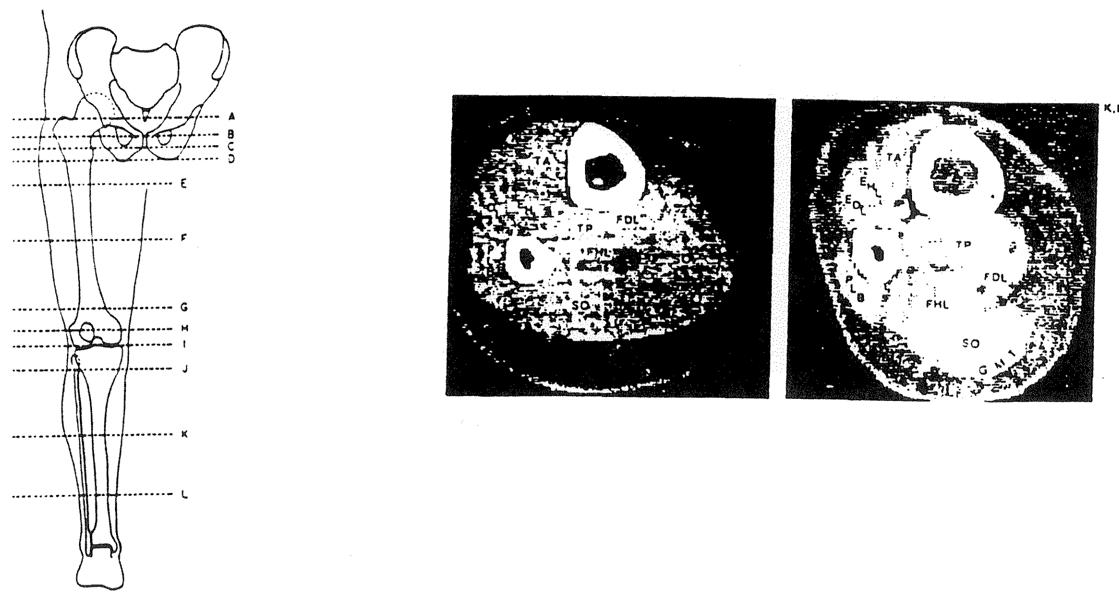

Figure 1. Tomograms of the lower leg.

The data were fabricated with a spreadsheet and edited via a C program, which stripped all non-printable characters and formatted the training set as pairs of the form:

(((x1 y1 Gray1) (BONE)) ((x2 y2 Gray2) (AIR))...)

where x1,y1, x2,y2,..., are pixel coordinate pairs, and where Gray1, Gray2,... are the gray scale values or linear attenuation values at the pixels.

The editing C program was also used to take in the one image created by with the spreadsheet and extrapolate additional training and testing data sets. Four 100x100 fabricated image matrices (TOMO1, TOMO2, ..., TOMO4.TRA) were extrapolated from the original image. The following data shows a smaller 2x4 image matrix (BONE1.TRA) manually fabricated from Table 1.

(((0 0 400) (Bone)) ((0 1 3000) (Bone)) ((0 2 –1000) (Air)) ((0 3 –1000) (Air)) ((1 0 –80) (Fat)) ((1 1 100) (Fat)) ((1 2 10) (Muscle)) ((1 3 80) (Muscle)))

Table 1. Training set configuration.

Row/Column	0	1	2	3
0	Bone	Bone	Air	Air
1	Fat	Fat	Muscle	Muscle

Bone:	*400 >= CT <=3000*
Fat:	*–80 >= CT <= –100*
Muscle:	*10 >= CT <= 80*
Air:	*CT <= –1000*

A 512x512 matrix was also created from the original 100x100 by padding fat tissue and air around the existing image. This image was created simply to evaluate performance on a large image once all training and testing was completed successfully. Besides the .TRA files described above, two other file structures had to be created for each tomogram:

1. ".PR" files (predict files) contain the input pairs only for each data set, that is:

   ((x1 y1 Gray1) (x2 y2 Gray2)...)

   This file format was used in the predicting or testing classification stage.

2. ".GDF" files (Graphic Display File) are created as the output disk file to be graphed to illustrate the results.

The final neural net design allows multiple layers and multiple nodes per layer. The algorithm used is as follows:

1. The number of units in the input layer matches the one important basis for the classification which is the Gray scale value or CT number. The number of output layers is defined by the number of possible outputs presented in the training stage. For example if the tomogram image matrix was as shown on Table 1, then the possible outputs are (BONE AIR FAT MUSCLE). That indicates 4 nodes in the output layer. To clarify, for a pair of the form (0 3 -1000) the desired output from the network should be (0 1 0 0). The second element on the list indicates 1 because (0 3 –1000) has a gray scale value of –1000 and that corresponds to air. The number of hidden layers and nodes per layer are variables which are prompted for at runtime. For simplicity this explanation will cover a three layer neural network which means one

hidden layer. The number of nodes on that layer was later optimized to six in the learning stage. We denote the activation levels of the units in the input layer by x_j, in the hidden layer by h_j, and in the output layer by o_j. Weights connecting the input layer to the hidden layer are denoted by $w1_{ij}$, where i indexes the input units (one) and j indexes the hidden units (six). Likewise, weights connecting the hidden layer to the output layer are denoted $w2_{ij}$, with i indexing the hidden layer units and j indexing the output layer units.

2. The weights are then initialized randomly. Originally the weights were randomly between –0.1 and 0.1.

$$w1_{ij} = \text{random}(-0.1,0.1),\ i = 0,\ldots,A,\ j = 1,\ldots,B$$

$$w2_{ij} = \text{random}(-0.1,0.1),\ i = 0,\ldots,B,\ j = 1,\ldots,C \tag{4}$$

3. The threshold units are initialized. The values of these threshold units never change.

$$x_0 = 1.0$$

$$h_0 = 1.0$$

4. An input/output pair is chosen from the training set. The input vector is denoted x_i and the target output vector is y_i.

5. The activations from the input layer are then propagated to the units in the hidden layer using the following sigmoid equation:

$$h_j = \frac{1}{1 + e^{-\sum_{i=0}^{256x256} w1_{ij}x_i}},\ \text{for all } j = 1,\ldots,B \tag{5}$$

6. The activations from the hidden layer units are propagated to the output layer units. The following equation was used to calculate the network's output:

$$o_j = \frac{1}{1 + e^{-\sum_{i=0}^{B} w2_{ij}h_i}},\ \text{for all } j = 1,\ldots,C \tag{6}$$

7. Errors of the output layer are computed based on the network's output (o_j) and the desired output (y_j).

$$\delta 2_j = o_j(1 - o_j)(y_j - o_j),\ \text{for all } j = 1,\ldots,C \tag{7}$$

8. Errors of the hidden layer are denoted $\delta 1$ and computed as follows:

$$\delta 1_j = h_j(1 - h_j)\sum_{i=1}^{C} \delta 2_i w2_{ji},\ \text{for all } j = 1,\ldots,B \tag{8}$$

9. The weights between the hidden layer and the output layer are adjusted based on the calculated errors. The learning factor, η, is a scale factor that tells the network how far to move in the direction of the gradient. A small η will lead to slower learning, but a

large η may cause a move through weight space that "overshoots" the solution vector (Hinton, 1992). During the training stage of this project, it was determined that η should be set to 0.25.

$$\Delta w2_{ij} = \eta\delta 2_j h_i, \text{ for } i = 0,\ldots,\text{B}, j = 1,\ldots\text{C} \qquad (9)$$

10. The weights between the input layer and the hidden layer are adjusted according to the following:

$$\Delta w1_{ij} = \eta\delta 1_j x_i, \text{ for } i = 0,\ldots\text{A}, j = 1,\ldots,\text{B} \qquad (10)$$

11. Process goes back to step 4. and repeats. When all the input pairs have been presented to the network, one epoch has been completed. Steps 4 through 10 are repeated until all network outputs match the desired output (100% learning rate) or until the maximum number of epochs is reached. The latter case represents non-convergence. In this case, the learning rate is calculated (<100%) in order to tabulate parameters and continue pursuit of the optimum network for convergence.

In order to speed up learning a momentum factor can be included. The weight update formulas then become:

$$\Delta w2_{ij}^{t+1} = \eta\delta 2_j h_i + \alpha\Delta w2_{ij}^{t}$$

$$\Delta w1_{ij}^{t+1} = \eta\delta 1_j x_i + \alpha\Delta w1_{ij}^{t} \qquad (11)$$

According to Minsky and Papert (1969):

> "Empirically, best results have come from letting alpha be zero for the first few training passes, then increasing it to 0.9 for the rest of the training. This process gives the algorithm some time to find a good general direction, and then moves it in that direction with some extra speed."

The final output from the program is a text file of the format:

```
(((x1 y1 gray) (predicted output, e.g., BONE))
((x2 y2 gray) (MUSCLE))...)
```

This output file is an ASCII file with a record length of 80. This file is then used as input to a FORTRAN graphing program using GKS to map each *i,j* to a *x,y* coordinate on the screen and gray to a color index.

A NEURAL NETWORK IMPLEMENTATION

This section describes the implementation of the architecture outlined above. The BONE identification program was written in Allegro Common LISP and implemented on a Sun SPARC 10 workstation. The program performs two major tasks: learning and predicting, as shown in the calling sequence below:

```
(Defun Bone ()
(Predict  (Learn  (Init-weights  (Train-set-in)))))
```

The learning stage is the process by which the neural network is presented with input pairs of pixels, CT values, and the expected tissue class for the pixels (e.g., "BONE"). The neural network uses the input pairs to determine a solution set of weights that fits all inputs with the expected outputs. Much like curve fitting of control points, the neural network finds the solution that fits all input pairs within selected criteria. Figure 2 shows a flow chart of the learning program. The convergence parameters η and α (see equations 9, 10, and 11) were experimentally determined based on learning performance. The initial weights were chosen randomly. Subsequent runs produced weight values approaching convergence. The final implementation uses theses values as the starting point. The goal of the learning stage design was to achieve an optimum learning rate with the smallest number of epochs. This means that the learning rate must correctly identify and classify all input pairs presented in the training set. The maximum number of passes or epochs can be changed to ensure the program converges.

Once the learning stage is completed successfully, the next stage is prediction. During this stage the learned knowledge is applied to classify tissues within the CT cross-section image slices. The implementation provides for classification of as many tissues as encountered in the training sets. The learned knowledge can be saved as a disk file so that expertise in classifying tissues can accumulate over the use of the application.

RESULTS AND DISCUSSION

Six fabricated tomograms where presented to the network. These tomograms were cyclically input to the network and iterated a number of times, until the weights stabilized. After this, tomograms containing unknown tissue types were classified using the trained network. All tomograms, at one point or another, serve as training or "unknown" set members. At first the number of hidden layers was set to two, as suggested by de Canete (1991). According to this reference, the number of nodes on the first layer is set to two (two to three times the number of units in the input vector). The second hidden layer was set to three times the number of nodes in the first hidden layer (six nodes). This proved to be an over-fitting choice. The network's performance is summarized in Table 2.

Table 2. Learning performance with two hidden layers.

Matrix Size	Network Architecture	Number of Epochs	CPU Time	Learning Rate
4x4	2:2:6	42	0.20 hrs	100.00
10x10	2:2:6	88	3.6 hrs	100.00
100x100	2:3:6	200+	55+ hrs	97.26

Over-fitting is used quite often with neural networks with the result of useful feature extraction. In this case, the CPU time required for convergence of a 100x100 matrix was very discouraging. The above results agree with the findings of Taylor (1991): "Too large a number of units in the network will give slow convergence, and may not generalize well; too few units will not allow a solution to the problem". At this stage the following actions were taken to speed up the learning stage:

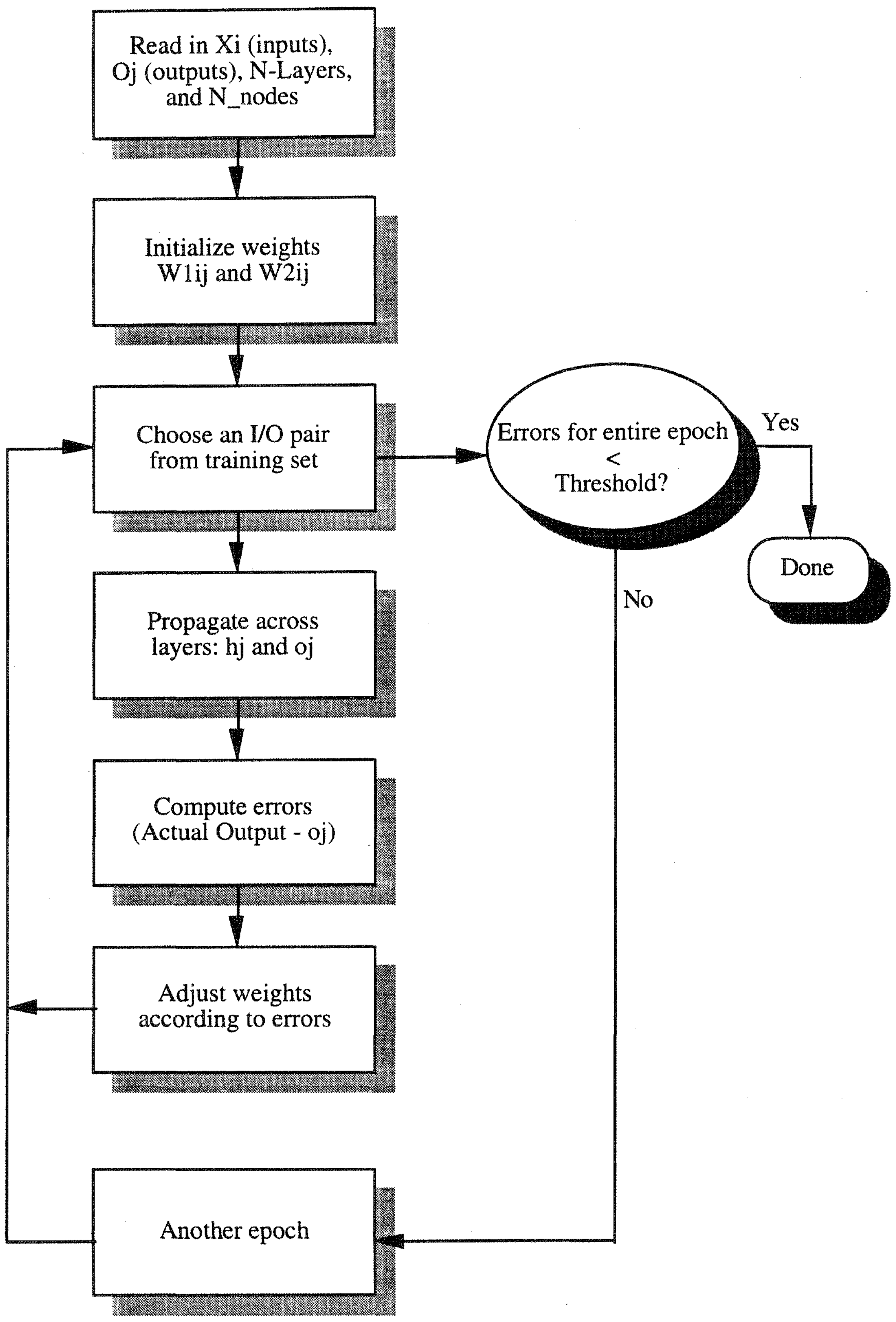
Read in Xi (inputs), Oj (outputs), N-Layers, and N_nodes
Initialize weights W1ij and W2ij
Choose an I/O pair from training set
Propagate across layers: hj and oj
Compute errors (Actual Output - oj)
Adjust weights according to errors
Another epoch
Errors for entire epoch < Threshold?
Yes
No
Done

Figure 2. Learning stage program flow chart.

- Each desired output and network output was thresholded around 0.1 and 0.9 instead of 0 and 1.

- The I/O method used to read in the training set was changed so that the entire set was read in with one disk I/O. The overhead of organizing pairs for network use was separated from the reading process. LISP I/O proved to be very slow compared to regular string manipulation.

- The number of hidden layers was reduced to one with six nodes. The learning factor was decreased to 0.25, and a momentum factor of 0.05 was added to epochs 30 through 50.

As a direct result of the above network architectural changes, the network performance summarized in Table 3 was achieved. This network architecture proved 100% efficient on both the learning and classification stages. These results are better than expected for typical neural networks. However, performance on real CT data may bring the performance down once noise affects the gray level values. The fabricated data is a perfect, noiseless case for the neural network since a solution plane exists for each tissue gray level value range.

Table 3. Classification accuracy of the final network architecture.

Image Size	Learn Rate	Learn Time	Number of Epochs	Network Config.	η	Gen. Rate	Gen. Time
10x10	100%	0.12 min	9	1:6	0.25	100%	0.67 min
20x20	100%	0.07 min	2	1:6	0.25	100%	0.03 min
100x100	100%	8 min	2	1:6	0.25	100%	3.5 min
100x100	100%	5 min	2	1:6	0.25	100%	3.0 min
100x100	100%	5.5 min	2	1:6	0.25	100%	3.2 min
512x512	100%	45 min	6	1:6	0.25	100%	11 min
100x100	TOMO2	------	-------	--------	-------	100%	3.5 min

As different combinations of tomograms for training and generalizing were presented to the network, a consistent behavior of repeatability was present. Given the same initial weight set, the network tends to converge in two epochs to 100%. The chosen learning and momentum factors proved to be optimal for the larger image matrices. As can be seen in Table 3, a larger number of epochs was required for image matrices smaller than 20x20. These factors will have to be adjusted when real CT data is presented to speed the learning and generalization stages.

FINAL RESULTS EVALUATION

Neural networks are a relatively new concept and their effectiveness in application to new fields is yet in research. This project attempts to prove a neural net's effectiveness as a tool in medical prosthesis and implant solid freeform fabrication. Evaluating the approach to the solution, the final network design proved flexible enough to adapt to the increasing possibilities of classes that may be encountered. The original design was expanded to allow variable hidden layers and variable nodes per layer. The final design was also not limited to a specific image matrix size. Additional design improvements include:

- the ability to recall existing knowledge from a disk file for future classification applications.

- the ability to generate graphical displays of the results.

Evaluating the results of this project has supported its original hypothesis that artificial neural networks are a reliable and automatic approach to tissue classification in tomograms.

CONCLUSION

This project has accomplished the following:

- A new approach to classifying tissue types by applying multi-layer perceptron principles has been used. This approach includes a creative method for encoding the tomogram image matrix, input and output, as well as the network's learned knowledge base.

- Performance of the network design was evaluated with sets of fabricated tomograms. The results of this evaluation demonstrate the effectiveness of the design and the approach experimentally.

Research efforts should be directed towards the performance evaluation of this network when presented with real CT data including tumors and unexpected tissues. Also, the network design should be expanded to include information about neighboring tissue types with the intent to automate the learning process, towards "unsupervised learning". We believe that by including information about neighboring tissue gray levels, the network will be able to discover regularities and relationships between the different parts of the input.

REFERENCES

1 Levy, R. A., Guduri, S., and Crawford, R., 1994, "Preliminary Experience with Selective Laser Sintering Models of the Human Temporal Bone", *American Journal of Neuroradiology*, Volume 15, pp. 473-477

2. Hinton, G. E., 1992, "How Neural Networks Learn Experience", *Scientific American*, Volume 267, No. 3, pp. 145.

3. Minsky, M., and Papert, S.,1969, *Perceptrons,* MIT Press, Cambridge, MA.

4. Taylor, J. G., 1991, "Simulated Ultrasound Tomographic Imaging of Defects", *Neural Network Applications*, Proceedings of the Second British Neural Network Society Meeting (NCM91), London, pp. 46.

5. de Canete, Frances, 1991, "Autonomous Controller Tuning by Using a Neural Network", *Artificial Neural Networks*, International Workshop, Granada Spain, Springer-Verlag Press.