Glove-Talk: A Neural Network Interface Between a Data-Glove and a Speech Synthesizer

S. Sidney Fels and Geoffrey E. Hinton

Abstract—**To illustrate the potential of multilayer neural networks for adaptive interfaces, we used a VPL Data-Glove connected to a DECtalk speech synthesizer via five neural networks to implement a hand-gesture to speech system. Using minor variations of the standard back-propagation learning procedure, the complex mapping of hand movements to speech is learned using data obtained from a single "speaker" in a simple training phase. With a 203 gesture-to-word vocabulary, the wrong is produced less than 1% of the time, and no word is produced about 5% of the time. Adaptive control of the speaking rate and word stress is also available. The training times and final performance speed are improved by using small, separate networks for each naturally defined subtask. The system demonstrates that neural networks can be used to develop the complex mappings required in a high bandwidth interface that adapts to the individual user.**

I. Introduction

ADAPTIVE interfaces are natural and important class of applications for neural networks. When a person must provide high bandwidth control of a complex physical device, a compatible mapping between the person's movements and the behavior of the device becomes crucial. With many devices the mapping is fixed and if a poor mapping is used, the device is difficult to control. Using adaptive neural networks, it may now be possible to build device interfaces where the mapping adapts automatically during a training phase. Such adaptive interfaces would simplify the process of designing a compatible mapping and would also allow the mapping to be tailored to each individual user. The key features of neural networks in the context of adaptive interfaces are the following:

- Neural networks learn input/output functions from examples provided by the user who demonstrates the input that should lead to a specified output. This "extensional" programming requires no computer expertise.
- Adapting the interface to the peculiarities of a new user is simple. The new user has only to create example data to retrain the network.
- Once trained, the networks run very quickly, even on a serial machine. Also, neural networks are inherently suitable for parallel computation.

Manuscript received July 8, 1991; revised February 14, 1992. This work was supported by the Institute for Robotics and Intelligent Systems, the Canadian Natural Science and Engineering Research Council, and the Ontario Information Technology Research Centre. The DECtalk speed synthesizer was provided by Terry Sejnowski and the Xerion neural network simulator used was written by Drew van Camp and Tony Plate.

The authors are with the Department of Computer Science, University of Toronto, Toronto, Canada, M5S 1A4.

IEEE Log Number 9201747.

II. Overview of the Glove-Talk System

To demonstrate the usefulness of neural networks for adaptive interfaces, we chose the task of mapping hand-gestures to speech [1]. The hand-gesture data is sensed by a VPL Data-Glove [2] that has two sensors for each finger. The sensors are fiber optic transducers which measure the finger flex angles. There is also a "polhemus" sensor attached to the back of the glove which measures the x, y, z, roll, pitch, and yaw of the hand relative to a fixed source. All 16 parameters are measured every 1/60th second. The speech synthesizer is a DECtalk model DTC01 from Digital Equipment Corporation. This synthesizer can perform text-to-speech synthesis and there is also user control of speaking rate and word stress.

The granularity of speech can be used to define a spectrum of possible methods for mapping from hand-gestures to speech. At the finest granularity, rapid finger movements could play the role of movements of the speech articulators, or they could represent some other parameterization of the speech wave such as the frequencies and amplitudes of the first four formants plus the pitch, the degree of voicing, and the amplitude of the nasal formant. This gives the user an unlimited vocabulary and analog control over the quality of the speech, but the finger movements must be extremely fast and they must be recognized very rapidly to produce real-time speech without a noticeable lag. In the middle of the spectrum, a brief movement or hand configuration could represent a diphone or syllable. At the other end of the spectrum, a complete hand-gesture could be mapped to a whole word without mapping temporal constituents of the gesture to temporal constituents of the word. This gives a fixed vocabulary which, like Chinese ideographs, makes it very arduous for the user to master a large vocabulary. However, for a small vocabulary the user can learn the task quite quickly. For this pilot study we chose to map complete hand-gestures to whole words.

Once trained, the Glove-Talk system works as follows: The user forms a hand-shape which represents a root word. Then s/he makes a movement forward and back in one of six directions. The direction chosen determines the word ending. The duration and magnitude of the gesture determine the speech rate and stress. The precise time at which the word is spoken is more complex. The user imagines that the end of the forward movement is like a button press that causes the word to be produced immediately, much like a conductor producing music. To avoid delays in producing the word, the shape of the hand is actually "read" near the beginning of the deceleration phase of the forward movement. This moment is called the "strobe-time" and is detected by the strobe

1045–9227/92$03.00 © 1992 IEEE

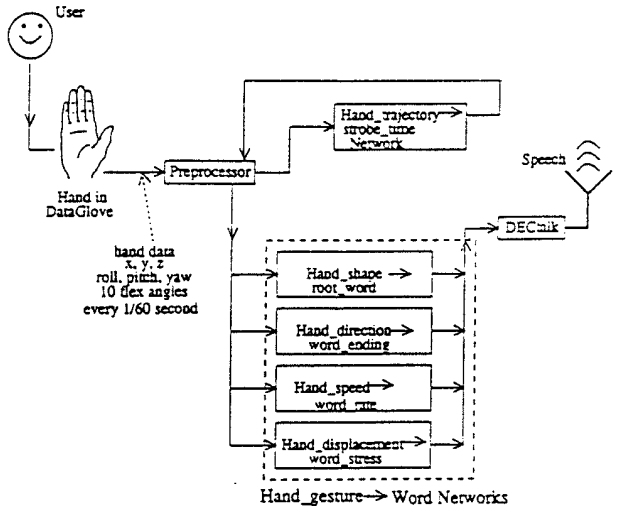

Fig. 1. Glove-Talk System.

network that continually monitors the directionless speed and acceleration of the whole hand using preprocessed information from the polhemus device. A block diagram of the Glove-Talk system is in Fig. 1.

When the strobe network detects an appropriate strobe-time, it sends a signal back to the preprocessor. The preprocessor then sends the appropriate buffered data to each of four neural networks to do the hand-gesture to word mapping. The root network determines the correct root word based on the static hand shape and orientation at the strobe-time. The ending network determines, form the direction of hand movement, which of the six possible endings (plural, -ed, -ly, -er, -ing, and normal) the user intended. The rate and stress networks determine the speaking rate and whether the word is stressed or not, based on the speed and magnitude of the hand movement, respectively. Once the mapping is completed, the appropriate commands (i.e., the word and the necessary rate and stress) are sent to the speech synthesizer which then speaks the word.

The current Glove-Talk vocabulary consists of 66 root words, each with up to six different endings. The total size of the vocabulary is 203 words. Five examples of the initial mapping of hand shapes (and orientations) to words are shown in Table I. Many of the hand shapes are derived from the American Sign Language (ASL) alphabet [3]. Orientation differences in the hand shapes are usually reserved for semantically opposite words; for example, the hand shapes for "come" and "go" have the same finger angles but are 180° of roll apart (i.e., "come' is made with the palm up and "go" is made with the palm down). The various endings of the words are formed by different directions of the hand movement. The mapping is given in Table II

Note that for some root words the endings do not correspond to the ones shown in Table II. For example, the "-s" ending of the root word "I" is "me". When any of the six endings do not exist for a root word, the normal ending is used. The 66 root words and six endings were extracted from the 850 word vocabulary of Basic English [4].

TABLE I
EXAMPLES OF GLOVE-TALK LANGUAGE

root word	hand shape
come	
go	
I	
you	
short	

TABLE II
DIRECTION OF MOVEMENT TO ENDING MAPPING

	Ending	Direction
1	normal	down
2	-s	up
3	-ed	towards user
4	-ing	away from user
5	-er	to user's right
6	-ly	to user's left

The following sections describe the five neural networks used. Full details of the complete system, the training data, and the performance can be found in [5].

III. LEARNING IN A MULTILAYER NEURAL NETWORK

All five neural networks were of the feedforward variety with one hidden layer. They were trained using backpropagation [6]. In a feedforward network, each unit has an activity level that is determined by the input received from units in the layer below. The total input, x_j, received by unit j is defined to be

$$x_j = \sum_i y_i w_{ji} - b_j \tag{1}$$

where y_i is the state of the ith (which is in a lower layer), w_{ji} is the weight on the connection from the ith to the jth unit and b_j is the bias of the jth unit. Biases can be viewed as the weights on extra input lines whose activity level is always one, so they can be learned in just the same way as the other weights. The lowest layer contains the input units and an external input vector is supplied to the network by clamping the states of these units. The state of any other unit in the network is determined by its activation function. One

example is a monotonic nonlinear function of its total input (sigmoid unit):

$$y_j = \frac{1}{1+e^{-x_j}}. \tag{2}$$

Other activation functions can also be used. For example, the activity levels of the output units of the root or ending networks represent probability distributions across mutually exclusive alternative. To ensure that these activity levels sum to one, the output value, y_i, of each output unit, i, is derived from the total input received by that unit, x_i, using the following nonlocal nonlinearity:

$$y_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3}$$

Many different error functions can be used. The most common is the sum-squared error which is simply the sum of the squared difference between actual and desired output activities. We use this error function for three of our networks, but for the root and ending networks we use a different error-function that is appropriate to the special behavior of the output units in these two networks (see (3)). The error function is:

$$E = -\sum_{j=1}^{n} d_j \log(y_j)$$
$$\text{where } d_j = \begin{cases} 1, & \text{for correct output unit} \\ 0, & \text{for incorrect output unit} \end{cases} \tag{4}$$

which simplifies to:

$$E = -\log y_j \tag{5}$$

where y_j is the activation of the correct output unit. The backpropagation procedure for deriving error-derivatives can easily be modified to use this new error function and the more complex behavior of the output units [7], [8]. In this case, the error derivatives with respect to the total input, x_j received by an output unit are obtained from (4) and (3), which are simply:

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j} = y_j - d_j \tag{6}$$

where d_j is the desired binary output value of unit j and y_j is its actual output.

Given the derivatives of the error function with respect to the weights, there are many different ways of training the networks. The weights can be updated "online" after each training case or the gradients can be accumulated over all training cases before doing a "batch" update. With batch updating, it is possible to significantly speed up the learning by using a line search and by using a method such as conjugate gradient to pick a better direction for the line search than the direction of steepest descent. However, since the networks we used were small, it was generally sufficient to use the batch version of steepest descent with a momentum term to accelerate the learning in ravines. For the somewhat larger root network we used a cross between batch and online methods (see the following)

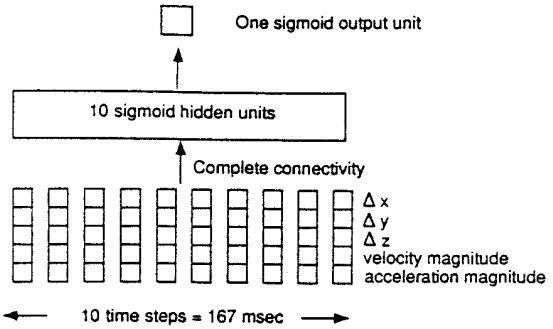

Fig. 2. The Strobe Network.

IV. The Hand Trajectory to Strobe Time Network

The network which we had most difficulty in designing and training was the strobe network (see Fig. 2) which must decide when the user intends to utter a word. This network segments the continuous stream of data from the Data-Glove, and, as is common in pattern recognition tasks, the segmentation turns out to be much harder than it appears.

The input data to the strobe network is a window of 10 time steps with five parameter at each time step. The first three parameters are Δx, Δy, Δz which are the difference between the current position and the previous position of the hand. The fourth parameter is the speed of the hand which is defined as $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$. The fifth parameter, which is loosely called the acceleration, is the current speed minus the previous speed. The speed and acceleration are the primary sources of information for detecting the beginning of the deceleration phase of the forward movement, but the Δx, Δy, Δz values are needed to allow the network to discriminate translational movements of the whole hand from translational movements of the polhemus caused by hand rotations.

The window size of 10 time steps (167 ms) is a compromise between ensuring that the network has enough information to detect a strobe-time and keeping the strobe network small. Notice that the strobe network must perform a forward propagation *every* time step to keep up with the Data-Glove. The 50 input units of the strobe network are fully connected to 10 hidden units which are all connected to a single sigmoid output unit which represents a binary decision about whether the most recent time is the right time to strobe (i.e., read) the hand shape (see Fig. 2). The input units act as a 10 time step shift register with five data values per time step.

The network was trained on 596 hand movement examples covering 27 907 time steps. The network training minimized the sum-squared difference between the actual and desired output values. Two variations of the target function were used to train the strobe net. In the first variation we hand-labeled appropriate strobe-times in the training data and required the network to give an output of one at these times and an output of zero at other times. For obvious reasons, this is called the delta target function. This makes life difficult for the network since our choice of strobe-time may differ by one time step from the network's natural choice, given the other training case it has to accommodate. In the second variation, each hand-labeled strobe-time was used to define the center of a Gaussian which

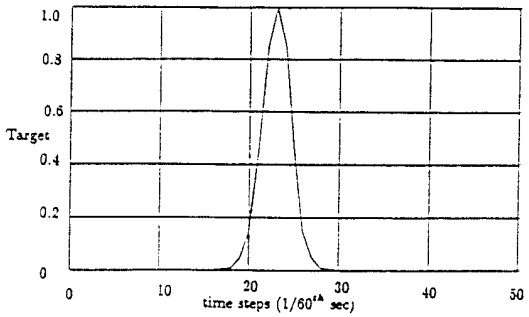

Fig. 3. Smooth target function for Strobe Network.

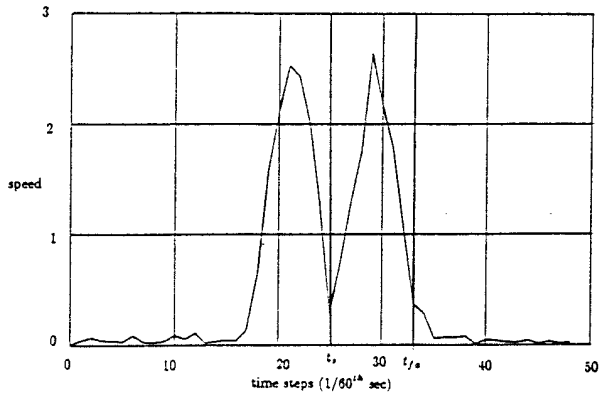

Fig. 4. Speed of hand versus time during hand movement.

specified the desired activity of the output unit. The Gaussian had a peak value of one and a standard deviation of one. This "smooth target function" (illustrated in Fig. 3) allows the strobe network some temporal latitude when recognizing the strobe time.

The performance of the strobe network is critical. If the network is to sensitive, extraneous words are produced and if it is too insensitive, many intended utterances will have to be retried. If the strobe-time is detected at the wrong time, the hand shape and direction of movement may be wrong. Also, the labeled training data for the other networks cannot be created until the strobe network is working . So it is very important to have a reliable, accurate strobe signal.

To illustrate what the strobe network recognizes, consider the typical time domain waveform of the speed of a user's hand movement in Fig. 4. In the first five time steps from rest, the hand accelerates and reaches its maximum speed. Then the hand begins to slow down in order to reverse direction. This takes about four time steps. The hand then accelerates and decelerates a it returns to its starting position. This takes another 10 time steps.

The strobe network is trained to recognize the initial acceleration followed by the deceleration. After detecting a strobe-time, the strobe network is not run for the next five steps. This refractory period eliminates double detections and frees the processor for simulating the other networks.

The hand-labeling of the training data for the strobe network is time-consuming. The training set was created in 21 sessions, where most sessions involved making 25 hand movements. The user made exactly the decided upon number of hand movements in each session, so the correct number of strobe-times in the continuous stream of measurements was known. Various hand shapes, hand movement directions, durations, and displacements were made to capture the variability in hand movements from different word contexts.

There were 27 907 input/output pairs in total and 596 of them were labeled as positive examples of strobe-times with the remainder being negative. The data was labeled by a human observer viewing the data graphically and marking the time of the first significant drop in speed of the user's hand after its initial acceleration. After the data had been labeled, it was possible to check the labeling accuracy by checking that the number of positive labels in each session was the same as the number of hand movements made. To save effort, data was labeled in a bootstrap manner. First, a small number of examples (approximately 60) were labeled by hand. These were then used to train the network. The remaining unlabeled training data were then forward propagated through the network. In most cases, some activity then occurred around the locations that would have been labeled by hand. Thus the initial, poorly trained network was used to focus the search for likely areas in the unlabeled training data where a strobe-time might occur.

To train the network using the delta target function required 1,325 sweeps through the entire training set with the weights being updated after each entire sweep. The learning rate was 0.001 and a momentum was 0.9 [6]. With the threshold set to 0.5, the total number of misses was 32 out of 596 and the total number of false alarms was 29 out of 27 311. Even though the number of training examples was very large, the time required to train the network on one processor of a Silicon Graphics 4D/240S was only a few hours due to the small network size. The strobe network has a 95% (190/200) hit rate on the test data set.

Using the smooth target function, 919 sweeps through the training set were necessary. The learning rate was 0.001 and a momentum of 0.75 was used. Allowing a tolerance of three time steps on either side of the hand-labeled strobe-time, the number of misses was 12 and the number of false-alarms was 21. When tested the strobe network achieves a 97% hit rate (194/200). As expected, this is better than the performance of the strobe network trained with the delta target function.

V. The Hand Shape to Root Word Network

The 16 input units of the root network represent the two flex angles of each finger (linearly scaled to lie between zero and one) and the sines and cosines of the roll, pitch, and yaw of the whole hand. Sines and cosines were used to eliminate the discontinuity that occurs when angles are represented by degrees. The input units are fully connected to 80 hidden units which, in turn, are fully connected to 66 output units (see Fig. 5). The network is trained to activate the appropriate output unit more than the others (see (3)). It requires 509 sweeps through the entire set of 8912 training examples to achieve a false alarm rate of 0.58% and a miss rate of 0.47% on the training data with a threshold of 0.5 (see Table III). The weights are updated every 66 examples (one

TABLE III
TRAINING AND TEST PERFORMANCE OF ROOT NETWORK

Data	Threshold	misses	false alarms	error (%)	disruptive error (%)	example set size
Training	.5	52	42	0.58	0.47	8,912
Testing	.5	37	28	1.70	1.29	2,178
Testing	.6	49	21	2.25	0.96	2,178

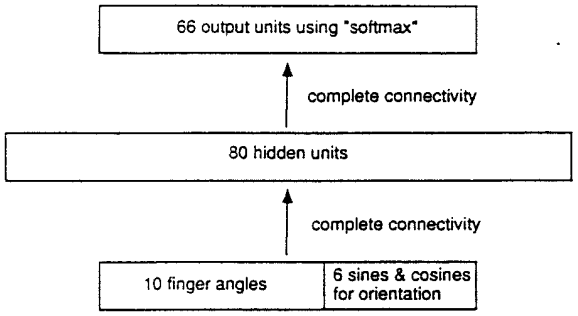

Fig. 5. The Root Network

per root word) using a learning rate of 0.01 and a momentum of 0.5. On test examples we tried two different thresholds to vary the miss/false-alarm trade-off. When the 0.5 threshold was used the number of false-alarms was 28 while the number of misses was 37. When we used a threshold of 0.6 the number of false-alarms decreased to 21 while the number of misses increased to 49. Considering that false-alarms result in wrong words being spoken whereas misses result in silence that can quickly be corrected by repeating the gesture, it is better to have fewer false-alarms even if this significantly increases the number of misses. The results for the training and test performance of the root network are summarized in Table III. More details of how the training data were collected and the effects of changing the architecture of the network or the input and output encodings are described in [5].

Once the strobe network is working it is simple to create training and test data for the other networks. Targets (combinations of the word root, ending, rate, and stress) are presented to the user and s/he simply makes the appropriate gesture. The strobe net recognizes when the gesture is made allowing the hand shape data to be recorded and labeled appropriately. A typical target presented to the user looks like this:

`shortly – fast (stressed)`

The user makes the hand shape representing the word "short" and quickly moves their hand to the far left and back again. Notice that each of the neural networks requires different data to be recorded. This fact was exploited by training the user on various parts of the target independently. Thus when getting training data for the root network the user focussed attention on getting the correct hand shape and paid less attention to the direction, speed, and displacement of the hand since those variations were only included to ensure that the recognition of the hand shape was robust against those variations. Similarly, when collecting training data for the other networks, attention was focused on the relevant aspect of the gesture.

The most difficult part of training for the user was remembering which hand shape represented each word. To ease the learning, training sessions were simple at first and progressively became harder. At first a fixed order of words was used and the user repeated each word 10 times in blocks of 10 words. The size of the blocks was increased as the user's accuracy and speed improved. Finally, session lengths of 500 randomly selected words were used. 7822 training examples were made using the fixed order scheme and 1090 training examples were created using the random scheme. In addition, 2178 test examples were created using the random scheme. As many incorrect hand shapes as possible were deleted before training and testing.

The training data was used to find the maximum and minimum values of the flex angles to allow linear scaling of the data. Note that the test data is scaled using the maximum and minimum values obtained from the *training* data. The data for the other networks were not scaled.

VI. ADAPTING TO A NEW USER

Once the strobe network is working, creating training and test data for the remaining neural networks is very simple. Data for each network can be obtained independently allowing the user to focus on one aspect of the input at a time. When the user is creating the training data, the idiosyncracies of his/her particular hand movements are recorded. When the networks are trained on this data they automatically adapt to the new user. We have not experimented with different users, but if users are fairly similar, learning should be very rapid if the weights learned from the previous user are used as the starting point for training on the data provided by the new user.

We have also considered the possibility of adapting the interface during normal use so that the way in which a user produces a word can gradually drift towards the simplest discriminable gesture. If the user indicated whenever the system made an error, the system would know the correct output for every case. For those cases in which it made an error or got the correct answer with low certainty, the system could then adapt appropriately. However, we have not yet experimented with this type of online adaptation so we do not know if simultaneous adaptation by the interface and the user will create difficulties for the user.

VII. THE NETWORKS FOR ENDING, RATE, AND STRESS

The role of the ending network is to translate the direction of movement of the user's hand to one of the six possible endings of the word. The six possible endings are mapped as shown in Table II.

We used a 30-10-6 back propagation network. The input is a 10 time step window of Δx, Δy, Δz values. The window includes the data at the strobe time and at the previous nine times. This size window is used because, occasionally, the strobe time is incorrectly detected during the backward hand movement. When this happens, the direction of motion at strobe time is the opposite of the intended forward direction, so the wrong ending is generated if only the last few time samples are used. This happened frequently using a window size of one. A larger window size retains the beginning motion of the user's hand, even when the detection of the strobe time is late.

There are six output units in the ending network, one per ending, and their activities are normalized using (3). 199 examples were used to train the network and 499 examples were used to test its performance. The network was trained for 125 sweeps through the training set with the learning rate (ϵ) at 0.05 and the momentum (α) equal to 0.5. After training the negative log error (see (4)) summed over all training cases was 1.15. There were no errors on the training data and only error on the test set which was probably caused by the user making the wrong movement.

The rate at which the word is spoken is determined by a 40-15-8 back propagation network which converts a 20 time-step window of directionless hand speed and directionless acceleration values to a real-valued speaking rate for each word. The window covers the strobe time and the previous 19 times. During training the user is asked to make movements which are very fast, fast, slow, or very slow to indicate four different speaking rates. The network adapts to the hand speeds that the user chooses.

The final output of the rate network is a speaking rate from 130–340 words per minute. The network has eight output units whose activities are in the range from 0 to 1. To get a real-valued rate from the output units we use a fixed post-processing operation. Each output unit is permanently associated with a particular speaking rate and the activity in the whole set of output units is then converted to a scaler (*rate*) using the following equation:

$$\text{rate} = \frac{\sum_i \mu_i o_i}{\sum_i o_i} \tag{7}$$

where μ_i is the speaking rate of unit i and o_i is its activity. The following values of μ_i were used: 60, 110, 160, 210, 260, 310, and 410 words per minute. The *rate* was restricted to lie between 130 and 340 words per minute.

It would be possible to train the rate network by simply backpropagating the derivative of the squared difference between desired and actual rates through this postprocessing function. However, this leads to strong interdependencies between the different output units, since they are all combined by the postprocessing. Also, it could lead to peculiar representations. For example, a slow rate could be represented by high activity in a very slow unit, slight activity in a very fast unit, and no activity elsewhere. We therefore used a different error function which specifies error derivatives for each output unit independently of the behavior of the other output units. We assume that each output unit should have a Gaussian tuning to speaking rate, so its desired activity is:

$$o_i = e^{-(r_d - \mu_i)^2/2\sigma^2} \tag{8}$$

where r_d is the desired speaking rate and σ is the same for all output units (a value of 25 was used). The output units used the sigmoid nonlinearity and they were trained to minimize the squared difference from their desired activities. The postprocessing function in (7) is still a sensible way to get a single scalar value since it corresponds to picking the speaking rate most likely to have produced the observed output activities under an appropriate Gaussian assumption.

The rate network was trained on 140 examples for 300 epochs. The learning parameters were: learning rate (ϵ) = 0.01 and the momentum (α) = 0.9. Since the speaking rate is a real value, the rate network does not have to be very accurate or reliable. As long as it provides a reasonable approximation to the desired rate, the speech will remain intelligible and its meaning will typically remain the same. For this reason we did not measure the performance of the rate network.

We noticed one major problem with the speaking rate when the system was running: the natural speed of hand movement differs greatly for different words. For example, to make a fast movement for the word "at" is much easier than for the word "righted." Given the current input data to the rate network, it is impossible for the network to know what word is being produced. So a user's fastest movement on the word "righted" and a slow movement of the word "at" may cause the same speaking rate even though the intended rate is different. Including context in the input data of the network could greatly reduce this problem. An obvious method would be to scale the output of the rate network by a different amount for each of the 203 words. These coefficients could easily be learned while training the rate network.

The stress network uses the displacement of the user's hand movement to decide whether to stress the word. It is a 40-5-1 back propagation network and receives exactly the same input as the rate network. It has a single sigmoid output unit whose value is thresholded at 0.5 to decide whether to stress the word. 100 sweeps through 140 examples were required to train the stress network using $\epsilon = 0.01$ and $\alpha = 0.9$. After training, the sum-squared error over all cases was 2.3 with four errors.

VIII. Achieving Real-Time Responses

The Glove-Talk system consists of three Unit processes. One handles the Data-glove communication and preprocessing, another runs the strobe network, and the last runs the remaining neural networks. The processes communicate using shared memory. When all the processes are running on a shared Silicon Graphics 4D/240S, the processing usually takes 10-20 ms. In addition to the computing delays, the response time is limited by the 33-ms delay in getting outputs from the Data-Glove. Using shared memory for the interprocess communication and synchronization provides close to real-time response.

To ensure a delay of only 20 ms in the four neural networks that identify the root word, the ending, the speech rate, and

the stress, requires about 0.8 million floating point operations per second because the four networks contain 8036 weights each of which requires a multiply and an add. The continually running strobe network is much less demanding because it is small. The delays in the Data-Glove, neural network, and speech synthesizer can be effectively eliminated by reading the hand-shape before the end of the forward movement of the hand.

IX. Concluding Remarks

Fairly rapid, intelligible speech is possible with the current Glove-Talk system. About 1% of the words spoken are incorrect, and about 5% of attempts result in no word being spoken due to failure to detect the gesture or failure to confidently identify the root word.

Obvious improvements include user control of pitch and loudness, continued "online" training while the system is in use, and a method for explicitly spelling out words that are not in the fixed vocabulary. One major improvement would be to increase the number of root words to the entire 850 words of Basic English. This may require substantial restructuring of the system, since static hand-shapes and orientations may not be sufficiently reproducible to allow 850 discriminable alternatives. It may therefore be necessary to use hand position or its temporal derivatives to distinguish between root words, or to encode root words by time-varying hand shapes. A sequence of two static hand-shapes, for example, would only require 30 discriminable alternatives to specify 900 root words, and a neural network should be good at adapting to coarticulation effects between the two consecutive hand-shapes.

It may prove necessary to abandon the use of static hand shapes altogether. We are currently investigating a much finer-grained mapping in which a neural net transforms the parameters measured by the data-glove into the frequencies and amplitudes of the first four formants plus the pitch, the degree of voicing, and the amplitude of the nasal formant. If the network can learn how to represent the rapidly changing formant parameters with smoothly changing hand parameters, and if a person can learn to produce the appropriate sequence of hand configurations, this system will have several advantages over the whole-word-method described in this paper. Although the initial learning will be much harder, the user will then have an unlimited vocabulary and much greater control of the speech.

References

[1] J. Kramer and L. Leifer, "The 'Talking Glove' A speaking aid for nonvocal deaf and deaf-blind individuals," in *Proc. RESNA 12th Ann. Conf.*, Louisiana, 1989, pp. 471–472.

[2] VPL Research Inc., *Data-Glove Model 2 Operating Manual*, CA, 1989.

[3] R.B. Wilbur, *American Sign Language and Sign Systems*. University Park Press, BA, 1979.

[4] C.K. Ogden, *Basic English: International Second Language*, prepared by E.C. Graham. New York: Harcourt, Brace & World Inc., 1968.

[5] S.S. Fels, *Building Adaptive Interfaces with Neural Networks: The Glove-Talk Pilot Study*, Tech. Rep. CRG-RT-90-1, University of Toronto, Toronto, Canada, 1990.

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[7] J.S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neuro-computing: Algorithms, Architectures and Applications*, F. Fougelman-Soulie and J. Herault, Springer-Verlag, 1989.

[8] G.E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185–234, 1989.

Sidney Fels received the B.A.Sc. degree in electrical engineering from the University of Waterloo in 1988 and the M.Sc. degree in computer science from the University of Toronto, Canada in 1990. He is currently a Ph.D. candidate in computer science at the University of Toronto.

His current research interests focus on neural network architectures for creating gesture to speech mappings.

Geoffrey E. Hinton received the B.A. degree in psychology from the University of Cambridge, U.K., in 1970 and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Scotland, in 1978.

He is a Professor in the Departments of Computer Science and Psychology at the University of Toronto, Canada, and a Fellow of the Canadian Institute for Advanced Research. His research interests include methods of using connectionist networks for learning, memory, perception, symbol processing, and motor control.