HIDDEN CONTROL NEURAL NETWORKS AND NEURAL PREDICTION MODELS FOR THE TASK OF SPEAKER VERIFICATION

Anthony Kelly and Eliathamby Ambikairajah

Speech Research Group
Department of Electronic Engineering
Regional Technical College, Athlone, Ireland

ABSTRACT – The Hidden Control Neural Network (Levin, 1990) and the Neural Prediction Model (Iso & Watanabe, 1990) were recently proposed as speech recognition models. The models are based on speech pattern prediction by multi-layer perceptrons. Each model was tested, by its proposer, with speaker independent digit recognition experiments. In both cases recognition accuracies in excess of 99% were achieved. This paper describes the use of both the Hidden Control Neural Network and the Neural Prediction Model to perform the task of speaker verification. The vulnerability of each model to changes in speech parameters over time is also investigated. A set of sixty utterances from one true-talker and six impostors, collected over a period of six months, is used to evaluate the speaker verification performance. The Neural Prediction Model based system yields a speaker verification accuracy of 100%, however, this falls to 90% for the Hidden Control Neural Network based system. Finally, a multi-transputer implementation of the Neural Prediction Model system is described. This system uses five transputers and operates in real-time.

1. INTRODUCTION

Multi-Layer Perceptrons (MLP's) are usually employed as pattern discriminators in speech applications. However, there are a number of practical problems in applying MLP's as discriminators. Firstly, the MLP discriminators require a very large training data set to learn complex discrimination hyperplanes for a large number of classes. Secondly, if new classes are added, discriminators in the system must be retrained using a data set for all classes. Finally, it is difficult to absorb temporal distortion of speech patterns and to deal with continuous speech.

Levin (1990) proposed a system, based on MLP frame predictors, which predicts speech patterns. The system is called a Hidden Control Neural Network (HCNN). Levin tested the HCNN with multi-speaker connected digit recognition experiments. A very high word accuracy of 99.3% was reported. Another system based on MLP frame predictors was proposed by Iso and Watanabe (1990). The system is called a Neural Prediction Model (NPM); it was tested with Japanese digit speech recognition. The results of these experiments showed that the NPM based speech recognition system outperformed conventional systems; a recognition error rate of only 0.2% was achieved.

A Neural Prediction Model to perform the task of speaker verification has been implemented by Ambikairajah and Kelly (1992). This paper compares a Hidden Control Neural Network based speaker verification system with the Neural Prediction Model based system.

Layered neural networks, like the MLP, are connectionist models that implement a non-linear parametric mapping from the input space to the output space. As such, these models are especially applicable for the approximation of non-linear multivariate functions. A network with one hidden layer of sigmoidal units can approximate, arbitrarily well, any continuous function. However, being a static model, a layered neural network is not capable of modelling systems with an inherent time variability, such as the speech production system. Both of the speaker verification systems described in this paper are based upon speech pattern prediction by MLP's. To overcome the problems associated with the time-varying nature of speech production, the neural network mapping must, somehow, change with time. The Hidden Control Neural Network and the Neural Prediction Model achieve this in different ways.

2. MODEL DESCRIPTIONS

For a particular speaker both the Hidden Control Neural Network and the Neural Prediction Model are constructed as a state transition network. For the HCNN each state is associated with a unique control

input, which can be seen as modulating the MLP mapping. For the NPM each state is associated with a different MLP predictor. The state transition configuration of these models is similar in form to the Hidden Markov Model.

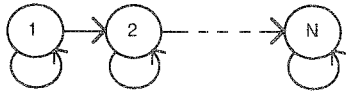

Figure 1. HCNN / NPM State Diagram.

Figure 1 shows a state diagram for what could be either an N state HCNN or an N state NPM. For the HCNN case N is the number of unique control symbols that are presented to the model, while in the NPM case N is the number of MLP's in the model.

2.1 Hidden Control Neural Network

A speaker model in HCNN is an MLP predictor. The MLP output is a frame prediction. The input consists of a number of previous frames along with a number of control inputs. The number of control inputs specifies the number of states that are in the HCNN model. Let frame vector t be represented by $\boldsymbol{a}_t$ and the HCNN prediction for frame vector t be denoted by $\boldsymbol{a}^*_t$. If S_t is the control symbol (Levin, 1990) associated with frame t and the frame prediction is based on two previous frames then $\boldsymbol{a}^*_t = f(\boldsymbol{a}_{t-1}, \boldsymbol{a}_{t-2}, S_t)$, where $f(x)$ is the MLP mapping. The difference between the predicted frame feature, $\boldsymbol{a}^*_t$, and the actual feature, $\boldsymbol{a}_t$, is defined as the prediction residual. This residual can be regarded as an error function for the model training, based on the back-propagation technique.

The set of control symbols is chosen such that each symbol is equidistant from each other symbol. For an N state HCNN there are N control inputs and the possible control symbols are as follows.

Q_1	=	1	0	0	...	0
Q_2	=	0	1	0	...	0
...	...	...	...	...	...	...
Q_N	=	0	0	0	...	1

The control symbol may be seen as modulating the MLP transformation, $f(x)$, such that it selects one of a set of possible invariant sub-transformations, $f_1(x), f_2(x), ..., f_N(x)$.

Figure 2 shows a HCNN system that bases its frame prediction on two previous frames.

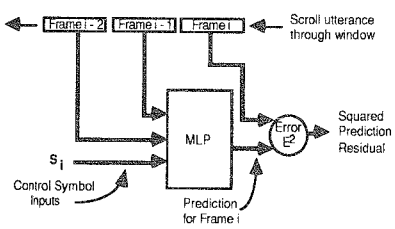

Figure 2. HCNN Model.

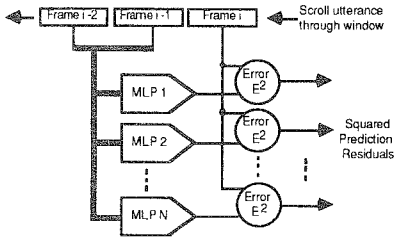

Figure 3. NPM Model.

2.2 Neural Prediction Model

The Neural Prediction Model consists of an array of MLP predictors. The hidden layer nodes have a sigmoidal transfer function, while the output nodes are linear. The MLP predictor outputs a predicted feature vector, $\boldsymbol{a}^*_t$, using the preceding speech feature vectors, $\boldsymbol{a}_{t-\tau}, ..., \boldsymbol{a}_{t-1}$, as inputs. The subscripts represent frame numbers of speech feature vectors. The symbol τ represents the number of input speech feature vectors used for prediction. The difference between the predicted frame

feature, $\boldsymbol{a}^*_t$, and the actual feature, $\boldsymbol{a}_t$, is defined as the prediction residual. This residual can be regarded as an error function for the MLP training, based on the back-propagation technique.

Figure 3 shows a Neural Prediction Model (Ambikairajah & Kelly, 1992) with N MLP predictors. In this case each predictor predicts a frame given two previous frames.

3. ALGORITHMS

The testing and training algorithms for both the HCNN and the NPM are essentially the same, as such, they will be treated together in this paper. The main component of these algorithms is the segmentation process. This process involves assigning each frame of an utterance to a particular model state such that the prediction residual for the utterance as a whole is minimised. Recall that a state in an NPM is associated with a particular MLP and that a state in a HCNN is associated with a particular control symbol.

3.1 Testing Algorithm

According to the algorithm described below, for the prediction of a test utterance by an N state HCNN or NPM, input speech is divided into N segments and the i-th segment ($1 \le i \le N$) is predicted by the model in state i. The optimal segmentation of the input speech is determined by minimising the accumulated prediction residual, D,

$$D = \min_{\{n(t)\}} \sum_{t=1}^{T} \| \boldsymbol{a}^*_t (n(t)) - \boldsymbol{a}_t \|^2,$$

where $\| \cdot \|$ is an Euclidean norm of a vector, T is the number of frames in the utterance and $\boldsymbol{a}^*_t(n)$ represents feature vector t, predicted by the model, either NPM or HCNN, in state n. For the NPM, $n(t)$ determines which MLP predictor is assigned to the prediction at frame t, for the HCNN, $n(t)$ determines which control symbol is used for the prediction at frame t. It must satisfy the following constraints:

$$n(1) = 1,$$
$$n(T) = N,$$
$$n(t) = n(t-1) \text{ or } n(t-1) + 1, \; (1 < t \le T).$$

Under these constraints the minimisation can be accomplished by use of dynamic-programming. The sequence $\{n(1), \ldots, n(T)\}$ represents the utterance trajectory through the model states.

Figure 4 shows a possible segmentation, $n(t)$, of a 10 frame utterance using a 4 state model, either HCNN or NPM.

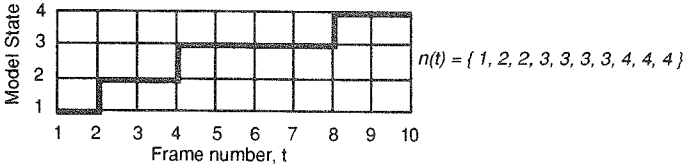

Figure 4. Example utterance segmentation with HCNN and NPM.

The process used to get the optimum segmentation, $n^*(t)$, is the same as that described by Ambikairajah and Kelly (1992). This process is based on dynamic programming and back-tracking techniques.

It was found that if the accumulated prediction residual is divided by the sum of the squares of each feature component in the utterance, then the resultant parameter, E, is the best one on which to base the speaker verification decision:

$$E = \frac{\sum_{t=1}^{T} (a_t - a^*_t)^2}{\sum_{t=1}^{T} \sum_{j=1}^{p} a_{tj}^2}.$$

In the above expression T is the number of frames in the utterance, a_t is frame t, a^*_t is the model prediction for frame t, component j of frame vector t is a_{tj} and p is the dimensionality of the feature vector.

3.2 Training Algorithm

The training goal of this type of system is to find a set of MLP predictor weights, which minimises the accumulated prediction residual for a training data set; for a speaker verification system the training data set is a collection of password utterances from a certain speaker. The objective function for the minimisation is defined as the average value for accumulated prediction residuals for all training utterances, D^*:

$$D^* = \frac{1}{M} \sum_{m=1}^{M} D(m),$$

where M is the number of training utterances, and $D(m)$ is the accumulated prediction residual for the m-th training utterance. The optimisation can be made by an iterative procedure, combining dynamic-programming and back-propagation techniques. The algorithm is as follows:

- initialise all MLP predictor weights to initial random values between zero and one (with the NPM there are many MLP's but with the HCNN there is only one MLP);
- set initial training utterance segmentations, $\{n^*(t)\}$, by following the dynamic-programming procedure described in the previous section;
- repeat the following procedures for all training utterances ($1 \leq m \leq M$) and until convergence;
- correct the MLP weights by back-propagation along the optimal trajectory $\{n^*(t)\}$, where desired output a_t is assigned to actual output $a^*_t(n)$ for the model in state $n^*(t)$;
- compute the accumulated prediction residual using dynamic-programming and determine the optimal trajectory $\{n^*(t)\}$ using its back-tracking.

4. EXPERIMENTS

This section describes two experiments, which are carried out to investigate the ability of Hidden Control Neural Networks and Neural Prediction Models to perform speaker verification. In each case, eight state models are used. That is, the HCNN's have eight distinct control inputs and the NPM's consist of eight MLP predictors. The password utterance is the phrase "SIX-THREE-NINE". It is spoken with a natural accent and the individual digits may be isolated or contiguous.

The time domain speech signal is band-pass filtered between 60 Hz and 3.4 kHz and sampled at 8 kHz. It is then analysed by a 32 ms frame window. As a feature vector for each frame the first 8 Mel Frequency Cepstral Coefficients (MFCC's) are extracted, (c_1, c_2, ..., c_8).

The first experiment is a preliminary investigation using a relatively small data set recorded at a single recording session. The results of this experiment give a general indication of how HCNN and NPM based speaker verification systems perform, but they give no indication of the systems' robustness to changes in people's speech parameters over time. To investigate this property a second experiment is carried out. In experiment two 60 true-talker utterances and 30 impostor utterances are collected. Half of the utterances are collected at one recording session and the other half are collected at a recording session six months later. The results of this experiment show whether or not the HCNN and NPM based speaker verification systems are vulnerable to changes in speech over time. In both experiments all of the utterances come from male speakers.

4.1 Experiment 1

Seventy password utterances are collected at one single recording session. The breakdown for the utterances of experiment one is as follows:

	Utterances	Training	Testing
True-Talker	50	40	10
Impostor	20	-	20

The impostor data set consists of the utterances of four separate people.

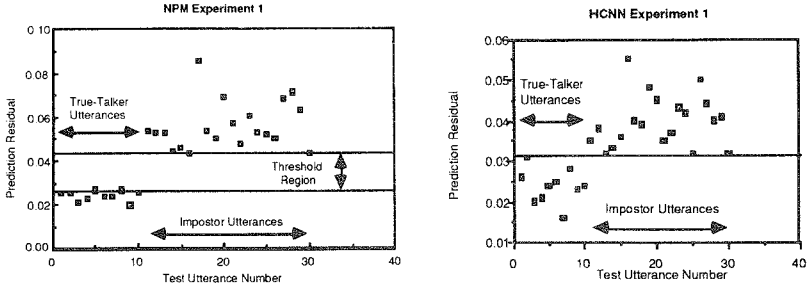

Figure 5. Test Results from Experiment 1.

Figure 5 shows the results of experiment one. It is seen that, in the NPM case, there is a large gap between the prediction residual levels for true-talker utterances and those for impostors, however, in the HCNN case the gap is almost non-existent. Nonetheless, in both cases it is possible to choose a threshold such that 100% speaker verification is possible with the test set. On account of the nature of the training data these systems could not be expected to perform well if the true-talker's speech changed much from the time of the recording session.

4.2 Experiment 2

One hundred password utterances are collected. Half of the utterances in each class are collected at one recording session and the other half are collected at a recording session six months later. The breakdown for the utterances of experiment two is as follows:

	Utterances	Training	Testing
True-Talker	70	40	30
Impostor	30	-	30

The impostor data set consists of the utterances of six separate people.

Figure 6 shows the results of experiment two. The gap that was present in the NPM case in experiment 1 has now disappeared. However, it is still possible to choose a threshold such that 100% speaker verification is achieved with the test set. This model is quite robust and unlike the model from experiment one it recognises true-talker utterances recorded at different times. The performance of the HCNN based speaker verification system has deteriorated even further, it is impossible to select a threshold that will result in 100% speaker verification with the test set. Indeed, it is seen that the best choice of threshold level results in a speaker verification accuracy of only 90% – six false acceptances. This reduction in performance, in the cases of both the NPM and the HCNN, is to be expected as in experiment two the models are trained to absorb the time-varying nature of the speaker's utterances as well as the unique speaker dependent aspects of the speaker's speech.

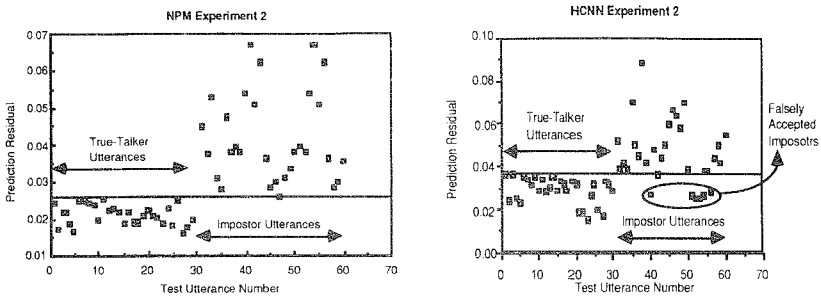

Figure 6. Test Results from Experiment 2.

Prior to training the feature vectors for each utterance are linearly scaled to lie in region between zero and one. This prevents numerical problems during training. A learn rate of 0.05 and a momentum term of 0.9 are used during training.

5. TRANSPUTER IMPLEMENTATION

Both of the speaker verification systems described in this paper are quite computationally intensive. Thus, without using a very high-performance computer, it is impossible to implement them in real-time. One solution to this is to implement the system using a parallel programming approach.

The Neural Prediction Model based system has been implemented on a transputer board, which consists of five T800 transputers. The implemented system reads a sampled utterance from disk and performs automatic end-pointing. The utterance is then analysed on a frame by frame basis and the feature vectors are extracted, this process easily lends itself to a parallel implementation on account of the repetitive nature of the frame by fame analysis. The NPM speaker verification algorithm, described in section 3, is then implemented. Finally, the prediction residual is compared with the pre-set threshold and the speaker verification decision is made. Figure 7 shows a process diagram for the transputer implemented NPM based speaker verification system.

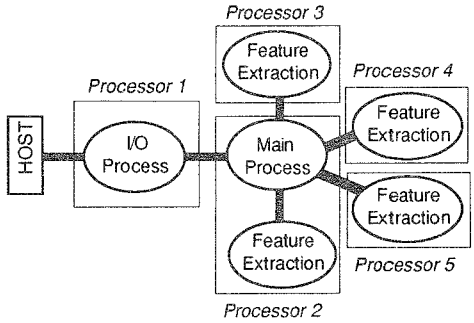

Figure 7. Process diagram for transputer implementation of NPM speaker verification system.

The total time from reading the speech samples to making the speaker verification decision varies between one and two seconds, depending on the length of the spoken password utterance. For this type of application this system can be considered to be a real-time implementation of a speaker verification system. The same system has also been implemented on a high-performance personal computer and the implementation time was found to be between 10 and 20 seconds, which cannot be considered a real-time implementation.