A Comparative Study on Kernel-Based Probabilistic Neural Networks for Speaker Verification *

K. K. Yiu, M. W. Mak, and S. Y. Kung

September 5, 2002

Abstract

This paper compares kernel-based probabilistic neural networks for speaker verification based on 138 speakers of the YOHO corpus. Experimental evaluations using probabilistic decision-based neural networks (PDBNNs), Gaussian mixture models (GMMs) and elliptical basis function networks (EBFNs) as speaker models were conducted. The original training algorithm of PDBNNs was also modified to make PDBNNs appropriate for speaker verification. Results show that the equal error rate obtained by PDBNNs and GMMs is less than that of EBFNs (0.33% vs. 0.48%), suggesting that GMM- and PDBNN-based speaker models outperform the EBFN ones. This work also finds that the globally supervised learning of PDBNNs is able to find decision thresholds that not only maintain the false acceptance rates to a low level but also reduce their variation, whereas the ad-hoc threshold-determination

*This work was supported by the Hong Kong Polytechnic University Grant Nos. G-W076 and A442. K. K. Yiu and M. W. Mak are with the Center for Multimedia Signal Processing, Dept. of Electronic & Information Engineering, The Hong Kong Polytechnic University. S. Y. Kung is with the Dept. of Electrical Engineering, Princeton University.

approach used by the EBFNs and GMMs causes a large variation in the error rates. This property makes the performance of PDBNN-based systems more predictable.

1 Introduction

Speaker verification aims to verify the validity of a claimed identity through voice. Text-dependent approaches, such as dynamic time warping (DTW) and hidden Markov models (HMMs) [1], explore the static and temporal characteristics of feature vectors. On the other hand, text-independent approaches, such as vector quantization (VQ) [2] and Gaussian mixture models (GMM) [3], assume independence among feature vectors and make use of distortion measures or probabilistic estimates. Early work that based on these approaches typically use data from the target speakers only to train the speaker models. As a result, discriminative information from non-target speakers (also known as anti-speakers or background speakers) will not be embedded in the speaker models.

Discriminative information can be utilized during model training and recognition. For the former, supervised learning algorithms are used to train a speaker model that discriminates within-class data from out-of-class data. For the latter, likelihood ratio [4] or scoring normalization [5] are applied during recognition.

Neural networks are one of the techniques that can embed discriminative information in the speaker models. For example, the elliptical basis function networks (EBFNs) proposed in [6] include the cluster centers of anti-speakers' speech in their hidden layer. It was shown that EBFNs outperform radial basis function networks (RBFNs) and VQ. The neural tree networks (NTNs) are another type of networks that use discriminative training, and research has shown that NTNs are superior to VQ in speaker recognition tasks [7].

Likelihood ratio [4] and scoring normalization that use cohort speakers [5] or the combination of cohort speakers and background speakers [8] have been applied to improve the performance of speaker verification systems. These scoring approaches achieve high

performance by constructing background speaker models and/or cohort models, which should accurately represent the characteristics of all possible impostors. Typically, the background models and speaker models are trained separately, which means that discriminative information is used during recognition rather than during training.

One of the main challenges in speaker recognition is to recognize speakers in adverse conditions. Noise is commonly considered as an additive component to the speech signals. Speaker models trained by using clean speech signals are usually subject to performance degradation in noisy environments. The present study compares the speaker verification performance of three kernel-based speaker models under clean and noisy environments. They are Gaussian Mixture Models (GMMs), Elliptical Basis Function Networks (EBFNs) and Probabilistic Decision-Based Neural Networks (PDBNNs) [9]. The comparison aims to demonstrate the effect of supervised learning on the speaker models (least squares learning on EBFNs and reinforced learning on PDBNNs). For example, by comparing GMMs against PDBNNs, the importance of reinforced learning can be highlighted.

Three problem sets have been used in this study. These include speaker verification experiments involving 138 speakers, speaker classification based on 2-D speech features and speaker verification using the noisy variants of the YOHO corpus. Through these problem sets, the modeling capability, robustness and generalization capability of the kernel-based speaker models are compared and investigated.

The key findings of this work are that (1) the GMMs and PDBNNs outperform the EBFNs in both clean and noisy environments and (2) the globally supervised training of PDBNNs is able to find decision thresholds that not only reduce the variation of false acceptance errors among all speakers in the system but also maintain the errors at very low levels.

The paper is organized as follows. In Sections 2.1 and 2.2, we briefly review the GMMs and EBFNs. Section 2.3 outlines the structural properties and learning rules of PDBNNs. The feature extraction method, speaker models, verification procedures, threshold determination procedures and evaluation results are explained in Section 3.

Finally, we conclude our discussions in Section 4.

2 Kernel-Based Probabilistic Neural Networks

2.1 Gaussian Mixture Models

Fig. 1 depicts the architecture of a classifier in which each class is represented by a Gaussian mixture model (GMM). GMMs make use of semi-parametric techniques for approximating probability density functions (pdf). As shown in Fig. 1, the output of a GMM is the weighted sum of R component densities. Given a set of N independent and identically distributed patterns $\mathcal{X}_i = \{x(t); t = 1, 2, \ldots, N\}$ associated with class ω_i, we assume that the class likelihood function $p(x(t)|\omega_i)$ for class ω_i is a mixture of Gaussian distributions, i.e.,

$$p(x(t)|\omega_i) = \sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \tag{1}$$

where $\Theta_{r|i}$ represents the parameters of the rth mixture component, R is the total number of mixture components, $p(x(t)|\omega_i, \Theta_{r|i}) \equiv \mathcal{N}(\mu_{r|i}, \Sigma_{r|i})$ is the probability density function of the rth component and $P(\Theta_{r|i}|\omega_i)$ is the prior probability (also called mixture coefficients) of the rth component. Typically, $\mathcal{N}(\mu_{r|i}, \Sigma_{r|i})$ is a Gaussian distribution with mean $\mu_{r|i}$ and covariance $\Sigma_{r|i}$.

The training of GMMs can be formulated as a maximum likelihood problem where the mean vectors $\{\mu_{r|i}\}$, covariance matrices $\{\Sigma_{r|i}\}$, and mixture coefficients $\{P(\Theta_{r|i}|\omega_i)\}$ are typically estimated by the expectation-maximization (EM) algorithm [10]. More specifically, the parameters of a GMM are estimated iteratively by [1]

$$\begin{aligned}
\mu_{r|i}^{(j+1)} &= \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t)) x(t)}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}, \\
\Sigma_{r|i}^{(j+1)} &= \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t)) \left[x(t) - \mu_{r|i}^{(j+1)}\right] \left[x(t) - \mu_{r|i}^{(j+1)}\right]^T}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}, \text{ and}
\end{aligned}$$

[1] To simplify the notation, we have dropped ω_i in Eqns (2) – (4).

$$P^{(j+1)}(\Theta_{r|i}) \quad = \quad \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}{N}, \tag{2}$$

where j denotes the iteration index and $P^{(j)}(\Theta_{r|i}|x(t))$ is the posterior probability of the rth mixture ($r = 1, \ldots, R$). The latter can be obtained by Bayes' theorem, yielding

$$P^{(j)}(\Theta_{r|i}|x(t)) \quad = \quad \frac{P^{(j)}(\Theta_{r|i})p^{(j)}(x(t)|\Theta_{r|i})}{\sum_{k=1}^{R} P^{(j)}(\Theta_{k|i})p^{(j)}(x(t)|\Theta_{k|i})} \tag{3}$$

in which

$$p^{(j)}(x(t)|\Theta_{r|i}) \quad = \quad \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{r|i}^{(j)}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left[x(t) - \mu_{r|i}^{(j)}\right]^T (\Sigma_{r|i}^{(j)})^{-1}\left[x(t) - \mu_{r|i}^{(j)}\right]\right\} \tag{4}$$

where D is the input dimension.

During recognition, a test vector $x(t)$ of an unknown class is fed to the GMMs, and the class index k is determined by the MAXNET:

$$k = \arg\max_{i=1}^{K} p(x(t)|\omega_i). \tag{5}$$

2.2 Elliptical Basis Function Networks

Elliptical basis function (EBF) networks [6] are a type of feedforward neural networks in which the hidden units evaluate the *distance* between the input vectors and a set of vectors called function centers or kernel centers, and the outputs are a linear combination of the hidden nodes' outputs. More specifically, the k-th network output has the form

$$y_k(x(t)) \quad = \quad w_{k0} + \sum_{j=1}^{M} w_{kj}\Phi_j(x(t)) \tag{6}$$

where

$$\Phi_j(x(t)) = \exp\left\{-\frac{1}{2\gamma_j}\left(x(t) - \mu_j\right)^T \mathbf{\Sigma}_j^{-1}\left(x(t) - \mu_j\right)\right\} \tag{7}$$

where μ_j and $\mathbf{\Sigma}_j$ are the function center (mean vector) and covariance matrix of the j-th basis function respectively, w_{k0} is a bias term, and γ_j is a smoothing parameter controlling the spread of the j-th basis function.

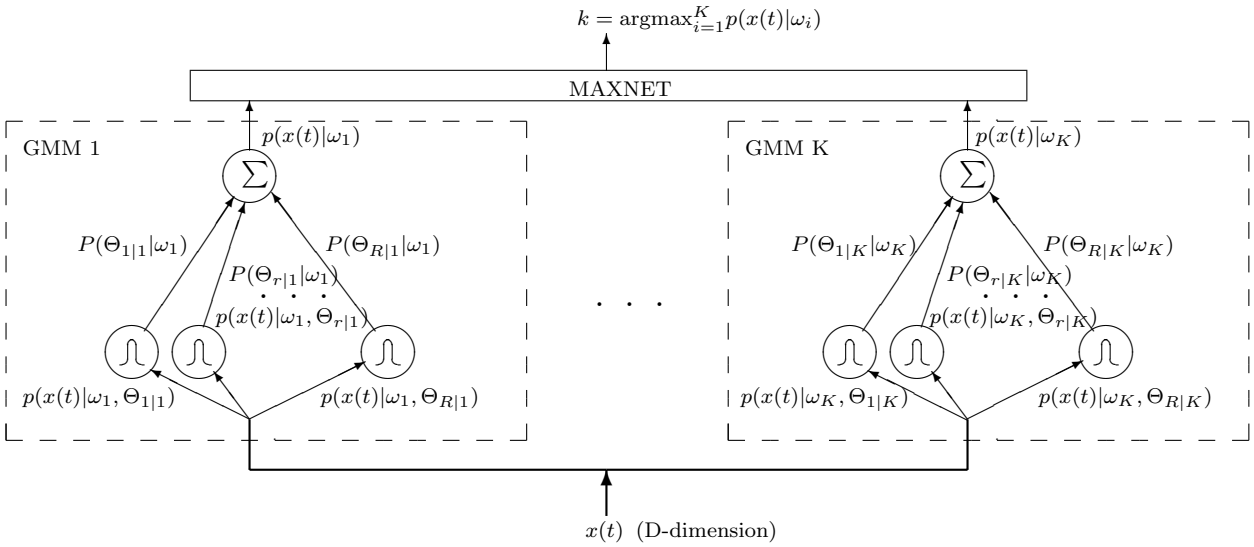


Figure 1: Architecture of a GMM-based classifier. Note that the classifier consists of K GMMs for a K-class classification task and that the "MAXNET" selects the GMM with the largest output as the identified class.

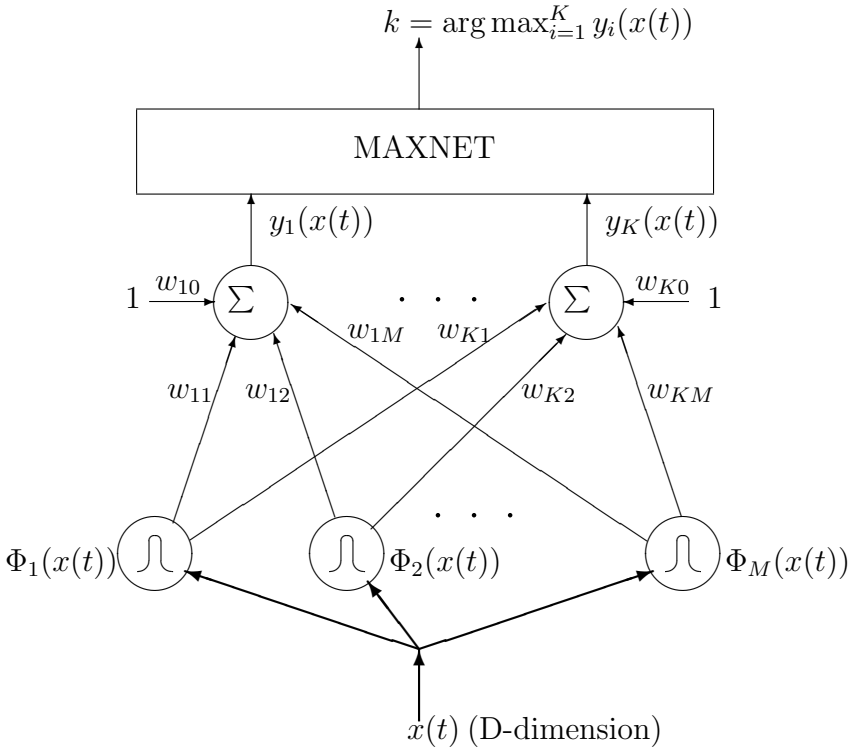


Figure 2: Architecture of a K-output EBF network.

Fig. 2 illustrates the architecture of an EBF network with D inputs, M function centers, and K outputs. It clearly shows that EBF networks have a three-layer architecture (input, hidden, and output layers). The input layer distributes the input patterns, $x(t)$, to the hidden layer. Each hidden unit is actually a Gaussian basis function with shape and location defined by a center μ_j and a covariance matrix $\mathbf{\Sigma}_j$. The number M of basis functions is typically much less than the number of training data.

The three-layer architecture with linear output units immediately leads to a two-stage training procedure. First, the kernel parameters $\{\mu_j,\ \mathbf{\Sigma}_j\}_{j=1}^{M}$ in the hidden layer are determined by fast unsupervised learning (such as k-means clustering and sample covariance). Second, the output weights $\{w_{kj}; k = 1, \ldots, K, j = 0, \ldots, M\}$ are determined by least squares techniques. Because of this two-stage training approach, EBF networks have a shorter training time than the backpropagation networks.

The kernel parameters can also be determined in an iterative fashion using the EM algorithm. For each iteration of EM, the mean vectors, covariance matrices, and mixture coefficients are updated according to Eqns. (2), (3) and (4) respectively. It has been shown that using the EM algorithm to estimate the kernel parameters produces superior networks than using the k-means algorithm and sample covariance [6].

We can observe the similarity between GMM-based and EBFN-based classifiers from their architecture (Figs. 1 and 2). For example, both of them compute the Mahalanobis distance (Eqns. (4) and (7)) between the input vectors and the kernel centers in the hidden layer. However, there are two important differences. First, a GMM computes the likelihood of observing the input vector $x(t)$, whereas an EBF network maps data from the input space to the output space. Second, the kernel parameters of a GMM must be estimated from data derived from its corresponding class. On the other hand, we apply data derived from all known classes (K classes in the case of Fig. 2) to estimate the kernel parameters of an EBF network. Even if we divide the EBF kernels into K groups and estimate the K sets of kernel parameters independently using the data derived from the K classes, EBF networks are still different from GMMs in that each of the EBF network's

outputs depends on the kernel outputs of the corresponding class as well as those from other classes. The output of a GMM ($p(x(t)|\omega_i)$), on the other hand, depends on the kernel outputs of its own class only. The consequent of this difference is that for EBF networks, discrimination among all the known classes is considered during the training phrase; whereas for GMMs this class discrimination is introduced during the recognition phase.

2.3 Probabilistic Decision-Based Neural Networks

Probabilistic decision-based neural networks (PDBNNs) [9] are a probabilistic variant of their predecessor, DBNNs [11]. Like DBNNs, PDBNNs employ a modular network structure as shown in Fig. 3. However, unlike DBNNs, they follow a probabilistic constraint. The subnet discriminant functions of a PDBNN are designed to model some log-likelihood functions of the form

$$\begin{aligned} \phi(x(t), \mathbf{w}_i) &= \log p(x(t)|\omega_i) \\ &= \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \right] \end{aligned} \tag{8}$$

where $\mathbf{w}_i \equiv \left\{ \mu_{r|i}, \Sigma_{r|i}, P(\Theta_{r|i}|\omega_i), T_i \right\}$ and T_i is the decision threshold of the subnet.

Learning in PDBNNs is divided into two phases: locally unsupervised (LU) and globally supervised (GS). In the LU learning phase, PDBNNs adopt the EM algorithm to maximize the likelihood function

$$\begin{aligned} l(\mathbf{w}_i; \mathcal{X}_i) &= \sum_{t=1}^{N} \log p(x(t)|\omega_i) \\ &= \sum_{t=1}^{N} \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \right] \end{aligned} \tag{9}$$

with respect to the parameters $\mu_{r|i}$, $\Sigma_{r|i}$, and $P(\Theta_{r|i}|\omega_i)$, where $\mathcal{X}_i = \{x(t); t = 1, 2, \ldots, N\}$ denotes the set of N independent and identically distributed training patterns from class i. The EM algorithm leads to an iterative update procedure identical to Eqns. (2), (3) and (4).

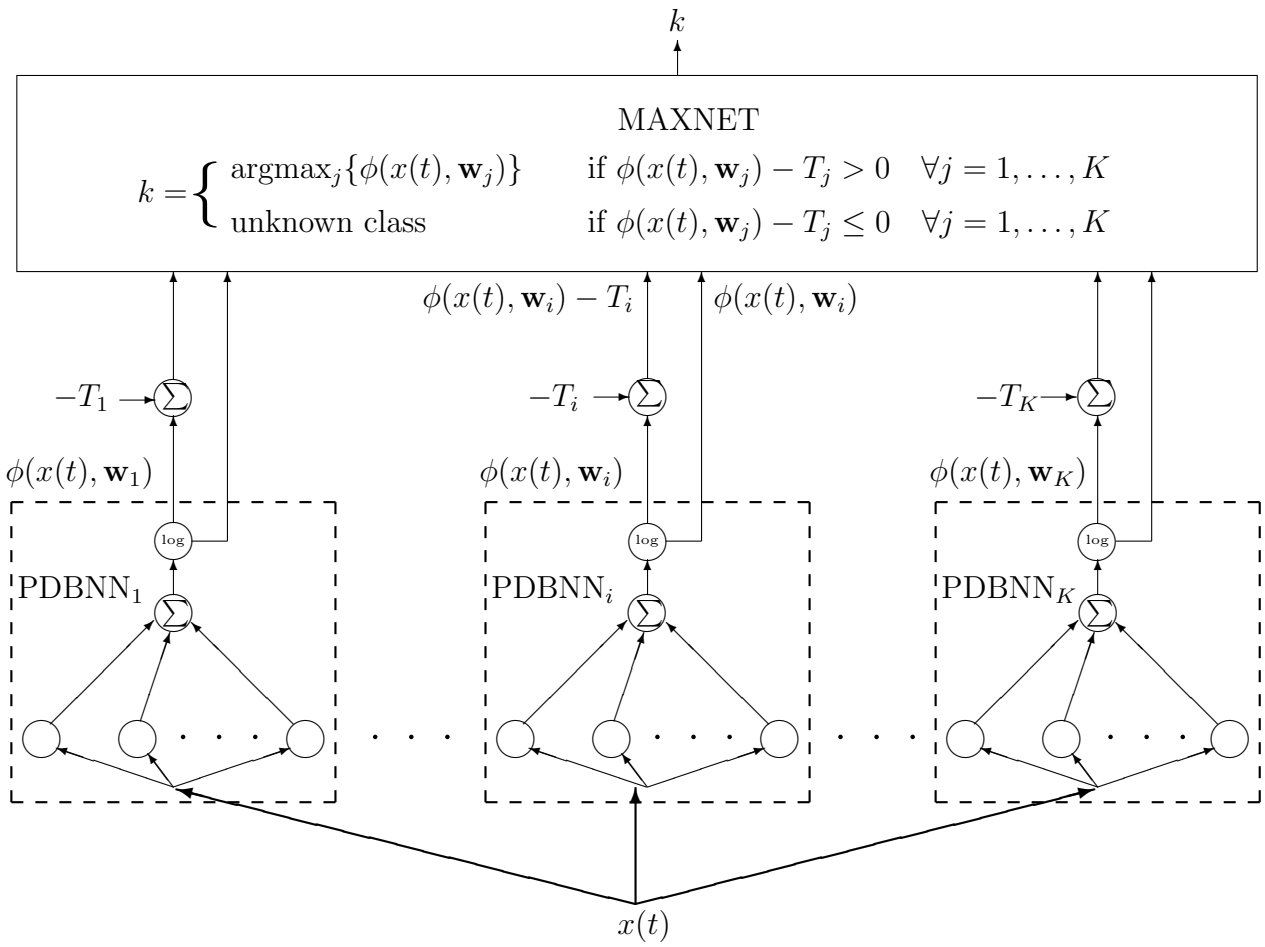


Figure 3: Structure of a PDBNN. Each class is modeled by a subnet. The subnet discriminant functions are designed to model the log-likelihood functions given by (8).

In the globally supervised (GS) training phase, target values are utilized to fine-tune the decision boundaries. Specifically, for any patterns $x(t)$ not belong to the i-th class but are misclassified to the i-th class, or for any patterns belong to the i-th class but are misclassified to another class, reinforced and/or anti-reinforced learning are applied to update the mean vectors and covariance matrices of subnet i. Thus we have

$$\begin{aligned}
\mu_{r|i}^{(j+1)} &= \mu_{r|i}^{(j)} + \eta_\mu \sum_{t,x(t)\in\mathcal{D}_2^i} h_{r|i}^{(j)}(t)\Sigma_{r|i}^{-1(j)}\left[x(t)-\mu_{r|i}^{(j)}\right] \\
&\quad -\eta_\mu \sum_{t,x(t)\in\mathcal{D}_3^i} h_{r|i}^{(j)}(t)\Sigma_{r|i}^{-1(j)}\left[x(t)-\mu_{r|i}^{(j)}\right] \\
\Sigma_{r|i}^{(j+1)} &= \Sigma_{r|i}^{(j)} + \frac{1}{2}\eta_\sigma \sum_{t,x(t)\in\mathcal{D}_2^i} h_{r|i}^{(j)}(t)\left(H_{r|i}^{(j)}(t)-\Sigma_{r|i}^{-1(j)}\right) \\
&\quad -\frac{1}{2}\eta_\sigma \sum_{t,x(t)\in\mathcal{D}_3^i} h_{r|i}^{(j)}(t)\left(H_{r|i}^{(j)}(t)-\Sigma_{r|i}^{-1(j)}\right) \qquad (10)
\end{aligned}$$

where $H_{r|i}^{(j)}(t) = \Sigma_{r|i}^{-1(j)}\left[x(t)-\mu_{r|i}^{(j)}\right]\left[x(t)-\mu_{r|i}^{(j)}\right]^T\Sigma_{r|i}^{-1(j)}$, $h_{r|i}^{(j)}(t)$ is the posterior probability identical to Eqn. (3), and η_μ and η_σ are user-assigned (positive) learning rates. The false rejection set $\mathcal{D}_2^i$ and the false acceptance set $\mathcal{D}_3^i$ are defined as:

- $D_2^i = \{x(t); x(t) \in \omega_i, \quad x(t) \text{ is misclassified to another class } \omega_j \}$,
- $D_3^i = \{x(t); x(t) \notin \omega_i, \ x(t) \text{ is classified to } \omega_i\}$.

An adaptive learning rule is employed to train the threshold T_i of subnet i. Specifically, the threshold T_i at iteration j is updated according to

$$T_i^{(j+1)} = \begin{cases} T_i^{(j)} - \eta_t l'(T_i^{(j)} - \phi(x(t), \mathbf{w}_i)) & \text{if } x(t) \in \omega_i \quad \text{(reinforced learning)} \\ T_i^{(j)} + \eta_t l'(\phi(x(t), \mathbf{w}_i) - T_i^{(j)}) & \text{if } x(t) \notin \omega_i \quad \text{(anti-reinforced learning)} \end{cases} \qquad (11)$$

where η_t is a positive learning parameter, $l(d) = \frac{1}{1+e^{-d}}$ is a penalty function, and $l'(d)$ is the derivative of the penalty function.

3 Applications to Speaker Verification

3.1 Speech Corpus

The YOHO corpus [12] was collected by ITT Defense Communication Division under a contract with the U.S. Department of Defense. It is a large-scale, scientifically controlled, and high-quality speech corpus for testing speaker verification systems at high confidence level. The corpus features "combination lock" phrases, 138 speakers (108 male, 30 female), inter-session variability, and high-quality telephone speech sampled at 8kHz with 16 bits per sample. The recording system of YOHO was set up in the corner of a large office. Low level noise could be heard from the adjoining offices. A handset containing an omni-directional electret microphone without noise-canceling features was used for recordings. For each speaker in the corpus, there are four enrollment sessions. Each of these sessions contains 24 utterances. Likewise, there are ten verification sessions for each speaker, with each session containing four utterances. Each utterance is composed of three 2-digit numbers (e.g. 34-52-67). The combination-lock phrases together with inter-session variability make YOHO ideal for speaker verification research.

In this work, all of the 138 speakers in the YOHO corpus have been used for experimental evaluations. Gaussian white noise with different noise power was also added to the clean YOHO corpus. Both the clean and noisy YOHO corpora were used in the evaluations.

TIMIT, which consists of 630 cooperative speakers (438 males, 192 females), is also commonly used for speech research. The TIMIT corpus is primarily designed for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. Although TIMIT has been used for studying text-independent speaker recognition [13], [6], it is not appropriate for evaluating speaker authentication systems due to its lack of inter-session variability. Therefore, we used the YOHO corpus in this study.

3.2 Feature Extraction

Speech signals can be considered as generated from quasi-stationary processes. Therefore, short-time spectral analysis can be applied to short speech segments, which results in a sequence of short-time spectra. In speech/speaker recognition, the short-time spectra are further transformed into features vectors. In addition to spectral analysis, many speech and speaker recognition systems use linear prediction (LP) analysis [14] to extract the feature vectors (known as the LP coefficients) from short segments of speech waveforms. One advantage of LP coefficients is that they can be computed efficiently. More importantly, the LP coefficients represent the spectral envelopes of speech signals (information about the formant frequencies and their bandwidth). The spectral envelopes are characterized by the vocal-tract resonance frequencies, vocal-tract length and spatially-varying cross-section areas. As all of these entities are known to be speaker-dependent, the LP coefficients are one of the candidate features for speaker recognition.

Several sets of features (such as LP coefficients, impulse responses, autocorrelation coefficients, cross-sectional areas and cepstral coefficients) can be derived from LP analysis. Of particular interest is that a simple and unique relationship exists among these features. Despite this simple relationship, it has been shown that the cepstral coefficients are the most effective feature for speaker recognition [15]. This is because the feature components of the cepstral coefficients are almost orthogonal. Based on the findings in [15], we used LP-derived cepstral coefficients in this study.

The feature extraction procedure is as follows. For each utterance, the silent regions were removed by a silent detection algorithm based on the energy and zero crossing rate of the signals. The remaining signals were pre-emphasized by a filter with transfer function $1 - 0.95z^{-1}$. Twelfth-order LP-derived cepstral coefficients were computed using a 28 ms Hamming window at a frame rate of 14 ms.

3.3 Enrollment Procedures

In the verification experiments, each registered speaker was represented by three different speaker models (GMM, EBFN, and PDBNN). A GMM-based speaker model consists of two GMMs, one representing the speaker himself/herself and the other representing all other speakers (called anti-speakers hereafter). An EBFN- or PDBNN-based speaker model consists of a single EBFN or PDBNN representing the corresponding speaker as well as all anti-speakers. For each registered speaker, all utterances in the four enrollment sessions corresponding to the speaker and a pre-defined set of anti-speakers (each speaker has his/her own set of anti-speakers) were used to train a speaker model. The speaker model was trained to recognize the speech derived from two classes—speaker class and anti-speaker class. To this end, two groups of kernel functions (one group representing the speaker himself/herself while the other representing the speakers in the anti-speaker class) were assigned to each speaker model. Hereafter, we denote the group corresponding to the speaker class as the speaker kernels and the one corresponding to the anti-speaker class as the anti-speaker kernels. For each registered speaker, a unique anti-speaker set containing 16 anti-speakers was created. Speech features derived from this set were subsequently used to estimate the anti-speaker kernels by using the EM algorithm. The anti-speaker kernels enable us to integrate scoring normalization [16] into the speaker models, which enhances the models' capability in discriminating the true speakers from the impostors.

Each of the GMMs and PDBNNs is composed of 12 inputs (12th-order cepstral coefficients were used as features), a pre-defined number of kernels, and one output. On the other hand, the EBFNs contain 12 inputs, a pre-defined number of kernels, and 2 outputs with each output representing one class (speaker class and anti-speaker class).

We applied the k-means algorithm to initialize the initial positions of the speaker kernels. Then, the kernels' covariance matrices were initialized by the K-nearest neighbors algorithm ($K = 2$). In other words, all off-diagonal elements were zero and the diagonal elements (being equal) of each matrix were set to the average Euclidean distance between the corresponding center and its K-nearest centers. The EM algorithm was subsequently

used to fine-tune the mean vectors, covariance matrices, and mixture coefficients (see Eqns. (2), (3) and (4)). All of the covariance matrices are diagonal. The same procedure was also applied to determine the mean vectors and covariance matrices of the anti-speaker kernels, using the speech data derived from the anti-speaker set. We found that initializing the mean vectors by k-means and the covariance matrices by K-NN reduces the number of EM iterations required to determine the maximum likelihood solution. As the k-means and K-NN algorithms run much faster than the EM algorithm, this approach can reduce the training time considerably.

The enrollment process for constructing a PDBNN-based speaker model involves two phases: locally unsupervised (LU) training and globally supervised (GS) training. The LU training phase is identical to the GMM training described above. In the GS training phase, the speaker's enrollment utterances and the utterances from all enrollment sessions of the anti-speakers were used to determine a decision threshold (see Section 3.5 below).

For the EBFN-based speaker models, the speaker kernels and anti-speaker kernels obtained from the GMM training described above were combined to form a hidden layer. In this work, γ_j in (7) was determined heuristically by

$$\gamma_j = \frac{9}{5} \sum_{k=1}^{5} \|\mu_k - \mu_j\| \tag{12}$$

where μ_k denotes the k-th nearest neighbor of μ_j in the Euclidean sense. We have empirically found that using five nearest centers and multiplying the resulting average distance by 9.0 give reasonably good results. However, no attempts have been made to optimize these values. Finally, singular value decomposition was applied to determine the output weights. Details of the enrollment procedure for EBFNs can be found in [6].

3.4 Verification Procedures

Verification was performed using each speaker in the YOHO corpus as a claimant, with 64 impostors being randomly selected from the remaining speakers (excluding the anti-speakers and the claimant) and rotating through all the speakers. For each claimant,

the feature vectors of the claimant's utterances from his/her 10 verification sessions in YOHO were concatenated to form a claimant sequence. Likewise, the feature vectors of the impostor's utterances were concatenated to form an impostor sequence.

3.4.1 Verification Procedures for PDBNNs and GMMs

For PDBNNs and GMMs, the following steps were performed during verification. The feature vectors from the claimant's speech $\mathcal{T}^c = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{T_c}\}$ was divided into a number of overlapping segments containing $T(< T_c)$ consecutive vectors as shown below[1]

$$\overbrace{\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4, \vec{x}_5, \vec{x}_6, \cdots\cdots, \vec{x}_T}^{\text{1st segment, } \tau_1}, \vec{x}_{T+1}, \vec{x}_{T+2}, \cdots, \vec{x}_{T_c}$$
$$\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4, \vec{x}_5, \overbrace{\vec{x}_6, \cdots\cdots, \vec{x}_{T+5}}^{\text{2nd segment, } \tau_2}, \vec{x}_{T+6}, \cdots, \vec{x}_{T_c}$$

For the t-th segment ($\mathcal{T}_t \subset \mathcal{T}^c$), the average normalized log-likelihood

$$z_t = \frac{1}{T} \sum_{\vec{x} \in \mathcal{T}_t} \{\phi_S(\vec{x}) - \phi_A(\vec{x})\} \tag{13}$$

of the PDBNN-based and GMM-based speaker models was computed, where $\phi_S(\vec{x})$ and $\phi_A(\vec{x})$ represents the log-likelihood function (Eqn. 9) of the speaker and anti-speaker respectively. Verification decisions were based on the criterion:

$$\text{If } z_t \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant} \end{cases} \tag{14}$$

where ζ is a speaker-dependent decision threshold (see Section 3.5 below for the procedure of determining ζ). A verification decision was made for each segment, with the error rate (either false acceptance or false rejection) being the proportion of incorrect verification decisions to the total number of decisions. In this work, T in Eqn. (13) was set to 500 (i.e., 7 seconds of speech), and each segment was separated by five vector positions. More

[1]The claimant can be the true speaker (in which case $\mathcal{T}^c$ represents a claimant sequence) or he/she can be an impostor (in which case $\mathcal{T}^c$ represents an impostor sequence).

specifically, the t-th segment contains the vectors

$$\mathcal{T}_t = \{\vec{x}_{5(t-1)+1}, \vec{x}_{5(t-1)+2}, \ldots, \vec{x}_{5(t-1)+500}\} \tag{15}$$

where $5(t-1)+500 < T_c$. Note that dividing the vector sequence into a number of segments has also been successful used in [3], [6] for increasing the number of decisions.

3.4.2 Verification Procedures for EBFNs

For the EBF-based speaker models, verification decisions were based on the difference between the scaled network outputs [6]. As the ratio of training vectors between the speaker class and the anti-speaker class is about 1 to 16, the networks favor the anti-speaker class during verification by always giving an output close to 1.0 for the anti-speaker class and close to 0.0 for the speaker class. This problem can be solved by scaling the outputs during verification so that the new average outputs are approximately equal to 0.5 for both classes. This can be achieved by dividing the output $y_k(\vec{x})$ by $2P(C_k)$, where $P(C_k)$ is the prior probability of class C_k. Specifically, we compute the scaled output $\tilde{y}_k(\vec{x}) = \frac{y_k(\vec{x})}{2P(C_k)}$ with $k = 1, 2$ so that $\frac{1}{N}\sum_{x\in\mathcal{X}} \tilde{y}_k(\vec{x}) \approx 0.5$, where N is the number of training vectors in the training set $\mathcal{X}$. A simple way to estimate the prior probability $P(C_k)$ is to divide the number of patterns in class C_k by the total number of patterns in the training set.

Similar to the PDBNN-based and GMM-based speaker models, we divided the claimant's utterance $\mathcal{T}^c$ into a number of overlapping segments. For each segment $\mathcal{T}_t$ (with segment length T), the scores

$$z_{t,k} = \frac{1}{T}\sum_{\vec{x}\in\mathcal{T}_t} \frac{\exp\{\tilde{y}_k(\vec{x})\}}{\sum_{r=1}^{2}\exp\{\tilde{y}_r(\vec{x})\}} \quad k = 1, 2 \tag{16}$$

corresponding to the speaker and anti-speaker classes were computed. Note that we have made use of the softmax function inside the summation of Eqn. (16). Its purpose is to ensure that $z_{t,k}$ is in the range $[0, 1]$ and that $\sum_k z_{t,k} = 1$, thereby preventing any extreme value of $\tilde{y}_k(\vec{x})$ from dominating the average outputs.

Verification decisions were based on the criterion:

$$\text{If } z_{t,1} - z_{t,2} \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant} \end{cases} \tag{17}$$

where $\zeta \in [-1, 1]$ is a speaker-dependent threshold (see Section 3.5 below) controlling the false rejection rate (FRR) and the false acceptance rate (FAR). Again, a verification decision was made for each segment (as defined in Eqn. (15)) at a rate of one decision per five feature vectors. Computing the difference between the two outputs is equivalent to normalizing the score in GMMs. Thus, we integrate scoring normalization into the network architecture.

In this work, equal error rate (EER)—false acceptance rate being equals to false rejection rate—was used as a performance index to compare the verification performance among different speaker models. As the speaker models remain fixed once they have been trained, EER can be used to compare the models' ability in discriminating the speaker features from the impostor features.

3.5 Determination of Decision Thresholds

As mentioned in Sections 3.3 and 3.4, we need to determine a decision threshold for each speaker model during enrollment. This threshold will be used during verification.

The procedures for determining the decision thresholds of PDBNNs, GMMs and EBFNs are different. For the GMM and EBFN speaker models, the utterances from all enrollment sessions of 16 randomly selected anti-speakers were used for threshold determination. Specifically, these utterances were concatenated and the procedure described in Section 3.4 was applied. The threshold ζ in Eqns. (14) and (17) was adjusted until the FAR fell below a pre-defined level. In this work, we set this level to 0.5%. The reason behind using anti-speakers' utterances rather than speaker's utterances is that it is much easier to collect the speech of a large number of anti-speakers. Hence, the thresholds obtained are more reliable than those would be obtained from speaker's speech. In addition,

using a pre-defined FAR to determine the decision thresholds enables us to predict the robustness of the verification system against impostor attacks [17].

To adopt PDBNNs to speaker verification and to determine the decision threshold of PDBNN-based speaker models, three modifications on the PDBNN's training algorithm have been made. First, the original PDBNNs use one threshold per sub-net. However, in our case, for each speaker we use one sub-net to model the speaker class and another one to model the anti-speaker class. To make PDBNNs applicable to speaker verification, we modified the likelihood computation such that only one threshold is required. Specifically, instead of comparing the subnet's log-likelihood against its corresponding threshold as in the original PDBNNs, we compared a normalized score against a single decision threshold as in Eqns. (13) and (14).

In the second modification, we changed the frequency at which the threshold is updated. The original PDBNN adopts the so-called batch-mode supervised learning. Our speaker verification procedure, however, is based on segmental learning (see Section 3.4). Hence, we modified the GS training to work on a segmental mode as follows. Let $\mathcal{T}_n$ be the n-th segment extracted from speaker's speech patterns $\mathcal{X}_S$ or from anti-speakers' speech patterns $\mathcal{X}_A$, the normalized segmental score is computed by evaluating

$$\begin{aligned} S(\mathcal{T}_n) &= S_S(\mathcal{T}_n) - S_A(\mathcal{T}_n) \\ &= \frac{1}{T}\sum_{\vec{x}\in\mathcal{T}_n}\phi_S(\vec{x}) - \frac{1}{T}\sum_{\vec{x}\in\mathcal{T}_n}\phi_A(\vec{x}) \\ &= \frac{1}{T}\sum_{\vec{x}\in\mathcal{T}_n}\{\phi_S(\vec{x}) - \phi_A(\vec{x})\} \end{aligned}$$

where $\phi_S(\vec{x})$ and $\phi_A(\vec{x})$ denotes the log-likelihood (Eqn. (8)) of speaker's speech and impostors' speech respectively. For each segment, a verification decision was made according to the criterion:

$$\text{If } S(\mathcal{T}_n) \begin{cases} > \zeta_{n-1}^{(j)} & \text{accept the claimant} \\ \leq \zeta_{n-1}^{(j)} & \text{reject the claimant} \end{cases} \tag{18}$$

where $\zeta_{n-1}^{(j)}$ is the decision threshold of the PDBNN-based speaker model after learning from segment $\mathcal{T}_{n-1}$ at epoch j. We adjusted $\zeta_{n-1}^{(j)}$ whenever misclassification occurs.

Specifically, we update $\zeta_{n-1}^{(j)}$ according to

$$\zeta_n^{(j)} = \begin{cases} \zeta_{n-1}^{(j)} - \eta_r l'(\zeta_{n-1}^{(j)} - S(\mathcal{T}_n)) & \text{if } \mathcal{T}_n \in \mathcal{X}_S \quad \text{and} \quad S(\mathcal{T}_n) < \zeta_{n-1}^{(j)} \\ \zeta_{n-1}^{(j)} + \eta_a l'(S(\mathcal{T}_n) - \zeta_{n-1}^{(j)}) & \text{if } \mathcal{T}_n \in \mathcal{X}_A \quad \text{and} \quad S(\mathcal{T}_n) \geq \zeta_{n-1}^{(j)} \end{cases} \tag{19}$$

where η_r and η_a are respectively the reinforced and anti-reinforced learning rates (more on next paragraph), $l(d) = \frac{1}{1+e^{-d}}$ is a penalty function, and $l'(d)$ is the derivative of $l(\cdot)$.

In the third modification, we introduced a new method to compute the learning rates. In the original PDBNNs, the learning rates for optimizing the thresholds are identical for both reinforced and anti-reinforced learning. However, in some situations, there may be many false acceptances and only a few false rejections (or vice versa), which means that anti-reinforced learning will occur more frequent than reinforced learning (or vice versa). In order to reduce the imbalance in the learning frequency, we make the reinforced (anti-reinforced) learning rate η_r (η_a) proportional to the rate of false rejections (acceptance) weighted by the total number of impostor (speaker) segments:

$$\eta_r = \frac{\text{FRR}^{(j-1)}}{\text{FAR}^{(j-1)} + \text{FRR}^{(j-1)}} \frac{N_{\text{imp}}}{N_{\text{imp}} + N_{\text{spk}}} \eta \tag{20}$$

$$\eta_a = \frac{\text{FAR}^{(j-1)}}{\text{FAR}^{(j-1)} + \text{FRR}^{(j-1)}} \frac{N_{\text{spk}}}{N_{\text{imp}} + N_{\text{spk}}} \eta \tag{21}$$

where $\text{FRR}^{(j-1)}$ and $\text{FAR}^{(j-1)}$ represent respectively the error rate of false rejections and false acceptances at epoch $j-1$, N_{imp} and N_{spk} represent respectively the total number of training segments from the impostors and the registered speaker, and η is a positive learning parameter. The first term of Eqns. (20) and (21) increases the learning rate if the corresponding error rate is large, which has the effect of rapidly reducing the corresponding error rate. The second term weights the learning rate according to the proportion of training segments in the opposite class, which has the effect of reducing the learning rate of the frequent learner and increasing the learning rate of the non-frequent learner. This arrangement can prevent the reinforced learning or the anti-reinforced learning from dominating the learning process and aims at increasing the convergence speed of the decision threshold.

In this work, we only modified the decision thresholds in the globally supervised training with the mean vectors $\mu_{r|i}$ and covariance matrices $\Sigma_{r|i}$ remain unchanged. This is because we want to keep $\mu_{r|i}$ and $\Sigma_{r|i}$ as the maximum likelihood estimates.

3.6 Pilot Experiments

The architecture of GMMs, EBFNs and PDBNNs depends on several free parameters, including the number of speaker kernels, the number of anti-speaker kernels, and the number of anti-speakers for finding the anti-speaker kernels. To determine these parameters, a series of pilot experiments involving 30 speakers from the YOHO corpus were performed. Equal error rates (EERs) were used as the performance indicator. To determine an appropriate number of speaker kernels, speaker models with different numbers of speaker kernels were constructed, and the numbers of anti-speaker kernels and anti-speakers were respectively fixed to 160 and 16. Table 1(a) summarizes the average EERs obtained by the GMM-based speaker models. Evidently, the EER decreases as the number of kernels increases. The decrease in EER becomes less significant after the number of speaker kernels reaches 40.

To determine an appropriate number of anti-speakers for determining the anti-kernels, we varied the number of anti-speakers while fixing the number of speaker kernels and anti-speaker kernels to 40 and 160, respectively. Table 1(b) summarizes the average EER obtained by the GMM-based speaker models. Optimal performance is obtained when the the number of anti-speakers is 32. In order to reduce processing time, we used 16 anti-speakers in the rest of the experiments.

We have also varied the number of anti-speaker kernels while fixing the number of speaker kernels and the number of anti-speakers to 40 and 16 respectively. Table 1(c) shows the average EER obtained by the GMM-based speaker models. The results show that no significant reduction in error rate can be achieved when the number of anti-speaker kernels reaches 160. Hence, 160 anti-speaker kernels were used in subsequent experiments.

Number of speaker's kernels	EER (%)
10	2.78
20	1.51
40	0.77
80	0.57
160	0.48

(a)

Number of Antispeakers	EER (%)
4	2.02
8	1.30
16	0.77
32	0.48
64	0.81

(b)

Number of Anti-speaker Kernels	EER (%)
40	0.83
80	0.83
160	0.77
320	0.75
640	0.79

(c)

Table 1: Average equal error rates based on 30 GMM-based speaker models with different numbers of (a) speaker kernels (where the number of anti-speakers and the number of anti-speaker kernels were set to 16 and 160 respectively), (b) anti-speakers (where the number of speaker kernels and anti-speaker kernels were set to 40 and 160 respectively), and (c) anti-speaker kernels (where the number of speaker kernels and anti-speakers were set to 40 and 16 respectively).

As the EBFNs, GMMs and PDBNNs use the same set of kernels, it is not necessary to repeat the above experiments for EBFNs and PDBNNs.

3.7 Large-Scale Experiments

Based on the results in Section 3.6, we set the number of speaker kernels and the number of anti-speaker kernels to 40 and 160 respectively, and for each speaker model we used 16 anti-speakers for determining the parameters in the anti-speaker kernels. Note that we have selected a sub-optimal number of anti-speakers in order to reduce the computation time in creating the speaker models.

In this paper, experimental evaluations on closed-set text-independent speaker verification based on all speakers (108 male, 30 female) in the YOHO corpus are presented. The aim is to evaluate the robustness of different pattern classifiers (speaker models) for speaker verification. Three kinds of pattern classifiers (GMMs, EBFNs and PDBNNs) were investigated in this study. To demonstrate the robustness of these classifiers, speech from the enrollment sessions of the YOHO corpus was used for training while the speech from the verification sessions was used for testing.

Table 2 summarizes the average FAR, FRR, and EER obtained by the PDBNN-, GMM- and EBFN-based speaker models. All results are based on the average of 138 speakers in the YOHO corpus. The results, in particular the EERs, demonstrate the superiority of the GMMs and PDBNNs over the EBFNs. The EER of GMMs and PDBNNs are the same since their kernel parameters are identical. Table 2 shows that the smallest EER obtained by the EBFN models is 0.48%, which is greater than 0.33%, the EER of GMMs and PDBNNs.

In terms of FAR and FRR, Table 2 demonstrates the superiority of the threshold determination procedure of PDBNNs. In particular, Table 2 clearly shows that the globally supervised learning of PDBNNs can maintain the average FAR at a very low level during verification, whereas the ad hoc approach used by the EBFNs and GMMs produces

Speaker Model	FAR (%)	FRR (%)	EER (%)
GMMs	8.01	0.08	0.33
EBFs	15.24	0.50	0.48
PDBNNs	1.10	1.87	0.33

Table 2: Average error rates achieved by the GMMs, EBFNs and PDBNNs based on 138 speakers in the YOHO corpus. The decision thresholds for GMMs and PDBNNs were determined by setting the pre-defined FAR to 0.5%; whereas, the thresholds for PDBNNs were determined by reinforced learning (see Section 3.5).

a much larger average FAR. Recall from our previous discussion that the pre-defined FAR for determining the decision thresholds of EBFNs and GMMs was set to 0.5%. The average FAR of EBFNs and GMMs are, however, very different from this value. This suggests that it may be difficult for us to predict the performance of the EBFNs and GMMs in detecting the impostor attacks.

Figure 4 depicts the FAR and FRR of individual speakers in the GMM-, EBFN- and PDBNN-based speaker verification systems. Evidently, most of the speakers in the PDBNN-based system exhibit a low FAR. On the other hand, the GMMs and EBFNs exhibit a much larger variation in FAR. We conjecture that the globally supervised learning in PDBNNs is able to find decision thresholds that minimize the variation in FAR.

Figure 5 shows the DET Curves [18] corresponding to Speaker 164 for different types of speaker models. In the DET plots, we use a nonlinear scale for both axes so that systems producing Gaussian distributed scores will be represented by straight lines. This property helps spread out the receiver operating characteristics (ROCs), making comparison of well-performed systems much easier. Note that the DET curves for the GMM and PDBNN are identical in this experiment because the globally supervised training updates the thresholds of PDBNNs only. It is evident from Figure 5 that the GMM- and PDBNN-based speaker models outperform the EBFN one.

3.8 Compared with Related Work

There are several speaker verification evaluations based on the YOHO corpus in the literature. For examples, Reynolds et al. [19] obtained a 0.51% EER and Higgins et al. [4] achieved a 1.7% EER. Both systems are based on GMMs. Note that the performance of our system (0.33% EER) is better that of [19] and [4]. However, these error rates can only be loosely compared with each other, since the evaluations were not performed under identical conditions (different training/testing paradigms and background speaker sets).

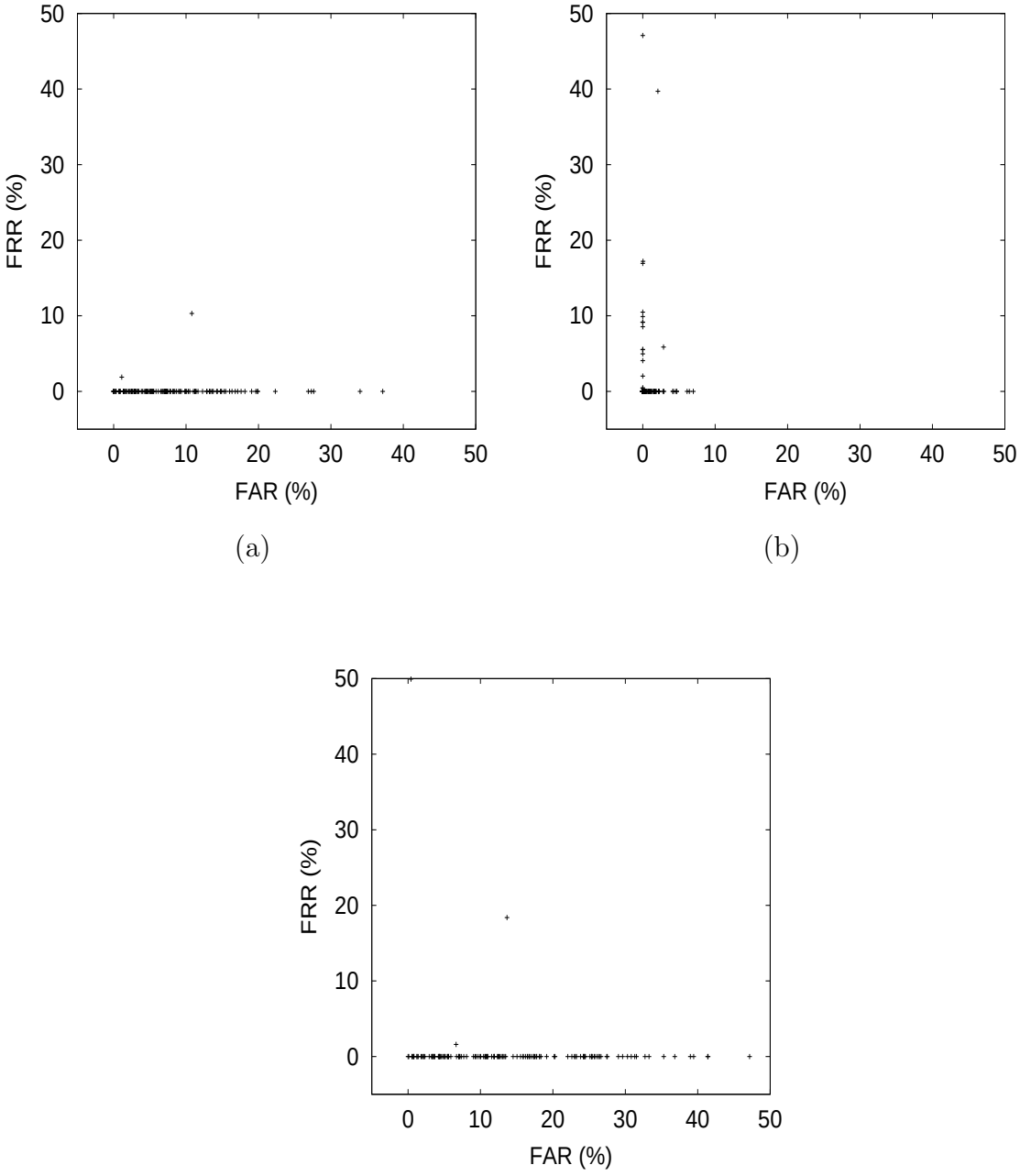
In order to compare with a more classical approach, we have repeated the same experiment using Vector Quantization (VQ) speaker models [2]. Using 64-center VQ speaker codebooks,[2] we obtained an EER of 1.29%. When the number of code vectors per VQ codebook reduces to 32, the EER increases to 1.61%. These EERs are much higher than that of the PDBNNs and GMMs.

It is also important to point out that EER is not the only criterion in judging speaker verification performance. In practical situations, we also need to consider the tradeoff between the false acceptance and false rejection. The key finding of this work is that the PDBNNs can effectively control this tradeoff through their threshold determination mechanism.

3.9 Decision Boundaries

To further illustrate the difference among the speaker models, we extracted the first and second cepstral coefficients of Speaker 162 and those of his anti-speakers and impostors to create a set of two-dimensional (2-D) speech data. Similar to the enrollment procedure in the speaker verification experiments, a PDBNN, a GMM and an EBFN (all with 2 inputs and 6 centers) were trained to classify the patterns into two classes. Therefore, except for the reduction in feature dimension, the training methods, learning rate and verification methods are identical to the speaker verification experiments described previously.

[2]The number of centers in VQ codebooks must be a power of 2.



(a) (b)

(c)

Figure 4: FRRs versus FARs (during verification) of 138 speakers using (a) GMMs, (b) PDBNNs and (c) EBFNs as speaker models.

	PDBNN/GMM		EBFN	
	Train	Test	Train	Test
EER(%)	4.12	24.61	6.86	27.17

Table 3: Performance of the PDBNN, GMM and EBFN in the 2-D speaker verification problem.

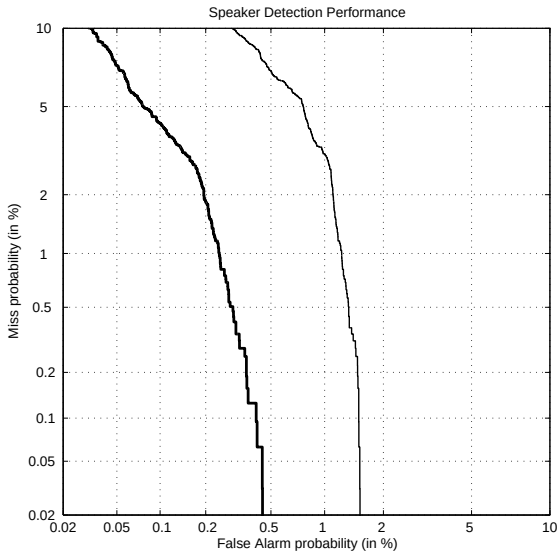


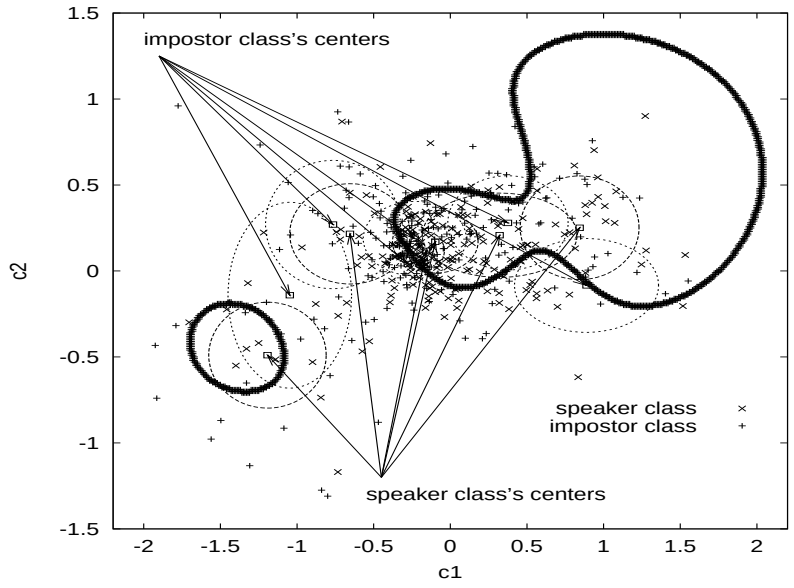
Figure 5: DET curves corresponding to Speaker 164. Thick curve: EBFN-based speaker model. Thin curve: GMM-based and PDBNN-based speaker models.

Table 3 compares the performance of the three speaker models, and Figure 6 shows the test data, decision boundaries, function centers, and contours of basis function outputs formed by these models. The decision boundaries are based on the equal error thresholds obtained from the data set. It is evident from Figure 6(a) that the decision boundaries formed by the EBFN enclose two regions, which belong to the speaker class, with a large amount of test data; whereas, the complement region, which belongs to the impostor class, extends to infinity. On the other hands, the decision boundaries created by the GMM and PDBNN extend to infinity in the feature space for both speaker class and impostor class. Both the decision boundaries (Figure 6) and the EER (Table 3), suggest that the GMM and PDBNN provide better generalization than the EBFN. These results also agree with what we have found in Table 2. The poor performance in EBFNs may be caused by the least squares approach to finding the output weights. As the EBFNs formulate the classification problem as a function interpolation problem (mapping from the feature space to 0.0 or 1.0), overfitting will easily occur if there are too many hidden nodes but too few training samples.

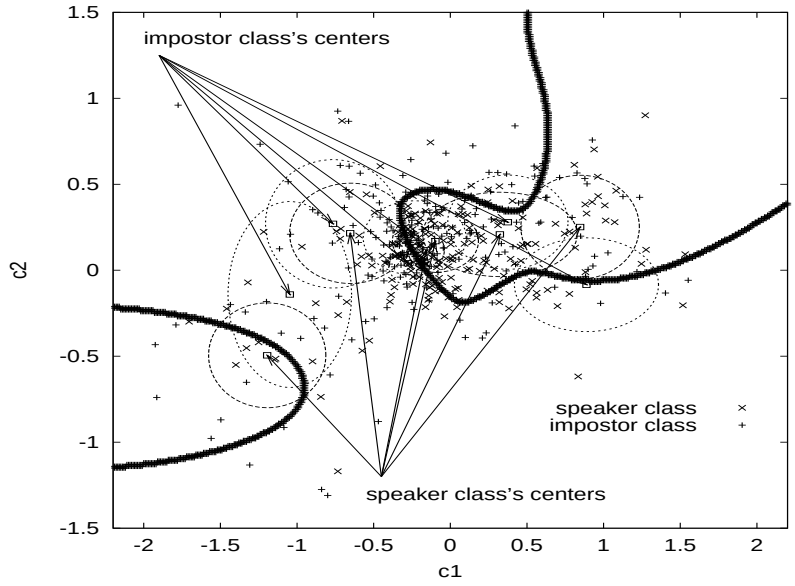
3.10 Robustness Against Noise

To test the robustness of different speaker models against noise, zero-mean Gaussian noise was added to the YOHO speech so that the resulting corrupted speech has an SNR of 10dB, 6dB, 3dB and 0dB. Segmentation files[3] derived from the clean YOHO corpus were used to specify the silence/speech regions of the corrupted YOHO speech. In practice, the same speech detector that we apply to YOHO speech should also be used to segment the corrupted YOHO speech. Our objective, however, is to compare the robustness of different speaker models against additive noise. Therefore, using the same segmentation files to define the speech regions of both clean and corrupted speech prevents the error introduced by the speech detector from interfering our comparison.

[3]These files only specify the silence and speech regions of the utterances. A speech detection algorithm based on the zero crossing rate and average amplitude was used to determine the speech regions.



(a)



(b)

Figure 6: Speaker verification problem using 2-D speech features. The figures depict the decision boundaries, function centers and contours of constant basis function outputs (thin ellipses) produced by (a) EBFNs and (b) GMMs and PDBNNs. Markers 'x' and '+' represent respectively the speaker's data and impostors' data.

Tables 4 summarize the average FAR, FRR, and EER obtained by the GMM-, PDBNN- and EBFN-based speaker models under different SNRs. The results show that the error rates of all models increase as the noise power increases. Such performance degradation is mainly caused by the mismatches in training and testing environments. Additive white noise contaminates the speech signals and therefore changes their acoustic characteristics. The speaker models, which were trained with clean speech, produced a reasonably low error rate for speech with a high SNR. However, their performance degraded rapidly when they were applied to noisy speech. Evidently, the EERs of PDBNNs and GMMs are smaller than that of the EBFNs under different SNR. Although PDBNNs and GMMs provide better generalization, the performance of PDBNNs and GMMs are still unacceptable at low SNR. In addition to additive noise, telephone speech may also be distorted by the handsets and the telephone channel. We are currently investigating compensation techniques [20] that aim to recover speech signals distorted by both additive and convolutive noise.

4 Conclusions

This paper addresses the problem of building a speaker verification system using kernel-based probabilistic neural networks. The modeling capability and robustness of these pattern classifiers are compared. Experimental results, based on 138 speakers and visualization of decision boundaries indicated that GMM- and PDBNN-based speaker models outperform the EBFN ones. Results also show that our modifications on the PDBNN's supervised learning not only make PDBNNs amenable to speaker verification tasks but also make their performance more predictable. This work also finds that PDBNNs and GMMs are more robust than EBFNs in recognizing speakers in noisy environments.

References

[1] C. Che and Q. Lin. Speaker recognition using HMM with experiments on the YOHO database. In *Eurospeech*, pages 625–628, 1995.

SNR	FAR (%)	FRR (%)	EER (%)
0 dB	43.98	55.47	34.00
3 dB	43.52	54.91	27.30
6 dB	42.51	53.59	20.32
10 dB	41.20	50.70	12.79
clean	8.01	0.08	0.33

(a)

SNR	FAR (%)	FRR (%)	EER (%)
0 dB	21.63	76.34	34.00
3 dB	19.52	77.53	27.30
6 dB	17.03	77.53	20.32
10 dB	13.67	76.38	12.79
clean	1.10	1.87	0.33

(b)

SNR	FAR (%)	FRR (%)	EER (%)
0 dB	30.95	66.57	37.51
3 dB	30.48	65.91	30.32
6 dB	29.97	65.16	22.45
10 dB	29.22	61.06	14.58
clean	15.24	0.50	0.48

(c)

Table 4: Average error rates obtained by the (a) GMM speaker models, (b) PDBNN speaker models, and (c) EBFN speaker models at different signal-to-noise ratios.

[2] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, and B. H. Juang. A vector quantization approach to speaker recognition. In *Proc. ICASSP 85*, pages 387–390, 1985.

[3] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans. on Speech and Audio Processing*, 3(1):72–83, 1995.

[4] A. Higgins, L. Bahler, and J. Porter. Speaker verification using randomized phrase prompting. *Digital Signal Processing*, 1:89–106, 1991.

[5] A. E. Rosenberg, J. Delong, C. H. Lee, B. H. Juang, and F. K. Soong. The use of cohort normalized scores for speaker verification. In *Proc. ICSLP 92*, pages 599–602, 1992.

[6] M.W. Mak and S.Y. Kung. Estimation of elliptical basis function parameters by the EM algorithms with application to speaker verification. *IEEE Trans. on Neural Networks*, 11(4):961–969, 2000.

[7] K. Farrell, S. Kosonocky, and R. Mammone. Neural tree network/vector quantization probability estimators for speaker recognition. In *Proc. Workshop on Neural Networks for Signal Processing*, pages 279–288, 1994.

[8] W. D. Zhang, M. W. Mak, and M. X. He. A two-stage scoring method combining world and cohort model for speaker verification. In *Proc. ICASSP'2000*, pages 1193–1196, June 2000.

[9] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. on Neural Networks, Special Issue on Biometric Identification*, 8(1):114–132, 1997.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of Royal Statistical Soc., Ser. B.*, 39(1):1–38, 1977.

[11] S. Y. Kung. *Digital Neural Networks*. Prentice Hall, New Jersey, 1993.

[12] Jr. J. P. Campbell. Testing with the YOHO CD-ROM voice verification corpus. In *ICASSP'95*, pages 341–344, 1995.

[13] D. A. Reynolds. Large population speaker identification using clean and telephone speech. *IEEE Signal Processing Letters*, 2(3):46–48, March 1995.

[14] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[15] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *J. Acoust. Soc. Am.*, pages 1304–1312, 1974.

[16] C. S. Liu, H. C. Wang, and C. H. Lee. Speaker verification using normalized log-likelihood score. *IEEE Trans on Speech and Audio Processing*, 4(1):56–60, 1996.

[17] W. D. Zhang, K. K. Yiu, M. W. Mak, C. K. Li, and M. X. He. A priori threshold determination for phrase-prompted speaker verification. In *Eurospeech'99*, volume 2, pages 1023–1026, 1999.

[18] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET Curve in assessment of detection task performance. In *Eurospeech'97*, pages 1895–1898, 1997.

[19] D. A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. In *Speech Communications*, pages 91–108, 1995.

[20] M. W. Mak and S. Y. Kung. Combining stochastic feautre transformation and handset identification for telephone-based speaker verification. In *Proc. ICASSP'2002*, pages I701–I704, 2002.