AN HIERARCHICAL EXEMPLAR-BASED SPARSE MODEL OF SPEECH, WITH AN APPLICATION TO ASR

Jort F. Gemmeke, Hugo Van hamme

Department ESAT, Katholieke Universiteit Leuven, Belgium
{jort.gemmeke,hugo.vanhamme}@esat.kuleuven.be

***Abstract*—We propose a hierarchical exemplar-based model of speech, as well as a new algorithm, to efficiently find sparse linear combinations of exemplars in dictionaries containing hundreds of thousands exemplars. We use a variant of hierarchical agglomerative clustering to find a hierarchy connecting all exemplars, so that each exemplar is a parent to two child nodes. We use a modified version of a multiplicative-updates based algorithm to find sparse representations starting from a small active set of exemplars from the dictionary. Namely, on each iteration we replace exemplars that have an increasing weight by their child-nodes. We illustrate the properties of the proposed method by investigating computational effort, accuracy of the eventual sparse representation and speech recognition accuracy on a digit recognition task.**

***Index Terms*—sparse representations, exemplars, hierarchy**

I. Introduction

For the last 20 years Automatic Speech Recognition (ASR) has been dominated by pattern recognition techniques based on Hidden Markov Models (HMMs) [1], employing Gaussian Mixture Models (GMMs) to model the acoustics. While GMMs allow for fast model training and scoring, training samples are pooled together for parameter estimation, resulting in a potential loss of the information that exists within individual training samples. A second issue with the use of GMMs is that their accuracy suffers when there is not sufficient training data to estimate all model parameters. Accordingly, there has been an increasing interest in the past decade in exemplar-based methods as a possible way to replace or augment GMM-based methods of speech processing [1], [2].

Exemplars (also known as *templates* or *episodes*) are records of detail of actual speech signals that encode idiosyncrasies such as the speaker and possibly even the context in which an utterance was produced. Recent advances in computing power, storage and the development of algorithms that can find structure in extremely large collections of observations, are now making exemplar-based approaches computationally feasible. Traditionally, exemplars have been used in combination with established techniques such as Dynamic Time Warping (DTW) [3], [4], Nearest Neighbour (NN or k-NN) classification [5] and Support Vector Machine (SVM) classification [6]. More recently, developments in fields such as Sparse Representations (SRs) and Compressed Sensing (CS) have revealed that speech may be effectively represented as sparse linear combinations of exemplars [7], [8], by finding the smallest number of exemplars in a very large collection of exemplars (a *dictionary*) that *jointly* approximate the observed speech token.

In recent work it has been shown the exemplar-based sparse representation model can be used for speech classification and recognition [9], [10]. Enforcing of sparsity in the linear combination of exemplars prevents overfitting [10] and enables applications in source separation and noise robustness, since noisy speech can be expressed as a sparse linear combination of noise and speech exemplars [8]. The fact that exemplars are combined linearly in SR-based ASR leads to a reduction in modelling error compared to NN using similarly sized dictionaries. Unfortunately, the computational cost of finding a sparse linear combination is much larger. Typically, exemplar-based sparse representation methods employ only hundreds to thousands of exemplars, rather than the hundreds of thousands or even millions of exemplars employed in DTW systems [2].

In this paper, we propose a hierarchical exemplar-based model of speech, as well as a new algorithm, to efficiently find sparse linear combinations of exemplars in dictionaries containing hundreds of thousands exemplars. First, we use a variant of hierarchical agglomerative clustering to find a hierarchy connecting all exemplars, so that each exemplar is a parent to two child nodes. Then, we use a modified version of a multiplicative-updates based algorithm[11], [12] to find sparse representations starting from a small active set of exemplars from the dictionary. Namely, on each iteration we replace exemplars that have an increasing weight by their child-nodes. We illustrate the properties of the proposed method by investigating computational effort, accuracy of the eventual sparse representation and speech recognition accuracy on a noise robust ASR task, AURORA-2.

II. Sparse Exemplar-based Representations of Speech

In our exemplar-based approach to ASR, speech signals are represented by their spectro-temporal distribution of acoustic energy, a *spectrogram*. The exemplar-based approaches proposed in this paper operate in the MEL-scale magnitude spectrogram domain, with the term *magnitude* referring to the square root of energy in a time-frequency coordinate.

The magnitude spectrogram describing a clean speech signal is a $B \times T$ dimensional matrix $\mathbf{S}$ (with B frequency bands and T time frames). To simplify the notation, the columns of this matrix are stacked into a single vector $\mathbf{s}$ of length $E = B \cdot T$.

We assume that an arbitrary speech spectrogram $\mathbf{s}$ can be expressed as a linear, non-negative combination of clean speech exemplars $\mathbf{a}^{\mathrm{s}}_j$, with $j = 1, \ldots, J$ denoting the exemplar index. These exemplars are magnitude spectrograms describing segments of speech signals extracted from a training database and are stacked in same way as was done to obtain $\mathbf{s}$. The superscript s denotes 'speech'; later in the paper we will be using the superscript n for noise. We write:

$$\mathbf{s} \approx \sum_{j=1}^{J} \mathbf{a}^{\mathrm{s}}_j x^s_j = \mathbf{A}^{\mathrm{s}}\mathbf{x}^{\mathrm{s}} \quad \text{subject to} \quad \mathbf{x}^{\mathrm{s}} \geq 0 \tag{1}$$

with x^s_j being the non-negative weight or *activation* of each exemplar. The J exemplars $\mathbf{a}^{\mathrm{s}}_1, \mathbf{a}^{\mathrm{s}}_2, \ldots, \mathbf{a}^{\mathrm{s}}_J$ are grouped into a speech exemplar *dictionary* $\mathbf{A}^{\mathrm{s}}$ as $\mathbf{A}^{\mathrm{s}} = [\mathbf{a}^{\mathrm{s}}_1 \ \mathbf{a}^{\mathrm{s}}_2 \ldots \mathbf{a}^{\mathrm{s}}_J]$ and the activations are stacked into $\mathbf{x}^{\mathrm{s}}$, a J-dimensional activation vector.

Previous research has shown that $\mathbf{x}^{\mathrm{s}}$ can be very *sparse* [13]. That is, only a few non-zero entries suffice to represent $\mathbf{s}$ with sufficient accuracy. In order to obtain $\mathbf{x}^{\mathrm{s}}$, we minimise the cost function:

$$d(\mathbf{s}, \mathbf{A}^{\mathrm{s}}\mathbf{x}^{\mathrm{s}}) + \lambda||\mathbf{x}^{\mathrm{s}}||_1 \quad \text{subject to} \quad \mathbf{x} \geq 0 \tag{2}$$

with distance function d and the second term a sparsity inducing L_1 norm of the activation vector controlled by λ. As a distance measure d we use the generalised Kullback-Leibler (KL) divergence. In previous work [8], a variant of Lee and Seungs iterative Non-negative matrix factorisation (NMF) algorithm [11] was used to obtain sparse representations:

$$\mathbf{x}^{\mathrm{s}}_{i+1} \leftarrow \mathbf{x}^{\mathrm{s}}_i . * (\mathbf{A}^{\mathrm{s}\mathrm{T}}(\mathbf{s}./(\mathbf{A}^{\mathrm{s}}\mathbf{x}^{\mathrm{s}}_i))) ./ (\mathbf{A}^{\mathrm{s}\mathrm{T}}\mathbf{1} + \lambda). \tag{3}$$

with $.*$ and $./$ denoting element-wise multiplication and division, respectively. The sparse representation $\mathbf{x}^{\mathrm{s}}_i$ is indexed by iteration counter $i \in [1, I]$. The vector $\mathbf{1}$ is an all-one vector of length E. On the first iteration, $\mathbf{x}^{\mathrm{s}}$ is initialised to unity.

III. The Hierarchical Exemplar-based Sparse Speech Model

A. Speech model

In this paper, we propose to accelerate the task of obtaining $\mathbf{x}^{\mathrm{s}}$ by imposing a hierarchical structure on the exemplars. The details of finding such a hierarchical structure are given in the next Section. For now, it suffices to say we hierarchically cluster all speech exemplars in the dictionary so that each exemplar is either a leaf node, or is a parent of two exemplars.

There has been some recent work on finding sparse representations using hierarchically structured dictionaries (cf. [14] and the references therein). In such work, however, a hierarchy is imposed on the sparsity pattern of the representation, in order to obtain more accurate sparse representations. In this paper we use a hierarchical structure to more efficiently obtain the sparse representation.

B. Splitting Multiplicative Updates Algorithm

In order to use the hierarchy to efficiently find sparse representations using large dictionaries, we exploit two properties of the approach described in Section II: First, the fact that (3) is an iterative method and second, the fact that only a few of the exemplars will have non-zero weight. The multiplicative update rule (3) is modified so the dictionary that is used in each iteration can change:

$$\mathbf{x}^{\mathrm{s}}_{i+1} \leftarrow \mathbf{x}^{\mathrm{s}}_i . * (\mathbf{A}^{\mathrm{s}\mathrm{T}}_i(\mathbf{s}./(\mathbf{A}^{\mathrm{s}}_i\mathbf{x}^{\mathrm{s}}_i))) ./ (\mathbf{A}^{\mathrm{s}\mathrm{T}}_i\mathbf{1} + \lambda). \tag{4}$$

The procedure is as follows. We start with an initial speech dictionary $\mathbf{A}^{\mathrm{s}}_0$ that is determined by taking the top $J_0 \ll J$ exemplars as determined by the hierarchy. Then, (4) is applied with this initial dictionary $\mathbf{A}^{\mathrm{s}}_0$ and initial sparse representation $\mathbf{x}^{\mathrm{s}}_0$ to obtain the sparse representation $\mathbf{x}^{\mathrm{s}}_1$. We search for exemplars that are *active* by determining the change in exemplar activation as $\mathbf{\Delta x}^{\mathrm{s}} = \mathbf{x}^{\mathrm{s}}_1 - \mathbf{x}^{\mathrm{s}}_0$. If for a certain exemplar it holds that $\Delta x^{\mathrm{s}} > 0$, and it is not a leaf node itself, its child exemplars are added from the dictionary $\mathbf{A}^{\mathrm{s}}$ to the dictionary $\mathbf{A}^{\mathrm{s}}_0$. Both the added exemplars get a weight $0.5 * x^s_1$, after which the parent exemplar is removed. After processing all activate exemplars in this fashion, the dictionary $\mathbf{A}^{\mathrm{s}}_1$ is obtained. This procedure is repeated on every subsequent iteration.

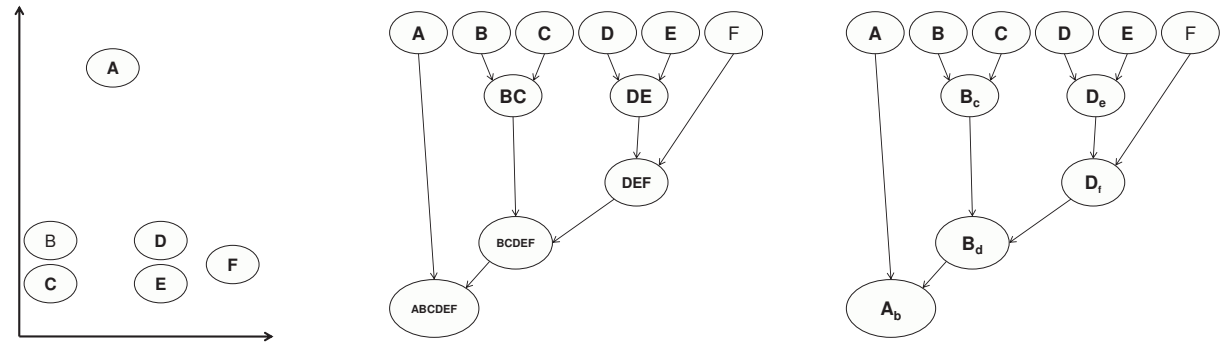
Since the the exemplar weights are sparse, only a limited number of exemplars will be active. This restrains the growth and eventual size of the active dictionary. Of course, different criteria for determining which exemplars are active can be defined.

C. Merging and pruning

Although not experimented with in this paper, we would like to point out that the proposed speech model and algorithm support two methods to constrain the size of the growing active dictionary: merging exemplars and pruning exemplars.

The merging method works by first finding inactive exemplars. Criteria for whether an exemplar is inactive could be a negative Δx^{s}, a small absolute weight $\mathbf{x}^{\mathrm{s}}$, or a combination of both. In order to merge two exemplars, it is necessary for both children of a parent node to be in the active dictionary, and both should be inactive. If this criterion is met, the two child nodes can be replaced by their parent node, which gets a weight equal to the sum of the weight of the child nodes. The merging and the splitting procedure are symmetric: for example if at a later iteration a previously merged exemplar obtains a $\Delta x^{\mathrm{s}} > 0$, it will be split again to restore the previous situation.

A possible downside of the merging method is that it may be difficult to have both child nodes inactive at once in the active dictionary, leading to only few merging actions in each iterations. An alternative is pruning: exemplars that are inactive can be removed from the active dictionary. The pruning can act on any node, be it a parent node or leaf node, but is irreversible: unlike in the merging operation, the child nodes of a node that has been pruned are not accessible any more in later iterations.



(a) Data points (b) HAC (c) HACAM

Fig. 1: Graphical representation of Hierarchical Agglomerative Clustering Around Medoids (HACAM). Figure (a) shows six datapoints in a two-dimensional space. Figure (b) shows the cluster tree resulting from Hierarchical Agglomerative Clustering (HAC). Figure (c) shows the cluster tree resulting from HACAM: in HACAM the parent node is one of the child nodes (the left node by default).

IV. Hierarchical clustering of exemplars

The exemplar-based speech model proposed above requires a hierarchical structure of the exemplar dictionary. To obtain such a hierarchy, we use a data-driven clustering method that is a variant of hierarchical agglomerative clustering (HAC) [15]. In HAC one iteratively merges the two closest exemplars until all exemplars are clustered. The difference with conventional HAC is that after merging two exemplars, we represent the parent node by one of the child nodes, rather than by the mean of the two exemplars. Rather than calculating which child node should be the representative node, we default to always use node with the lowest index in the dictionary (which is an arbitrary but consistent choice). We will refer to this approach, which is reminiscent of K-medoid clustering, as hierarchical agglomerative clustering around medoids (HACAM). A graphical example of HACAM is given in Figure 1.

The use of HACAM to obtain a hierarchical description of speech has the following benefits over other clustering techniques, such as HAC or divisive clustering:

- In this paper, every exemplar is associated with a label that describes the underlying speech class. HACAM, however, is data driven, and thus requires no knowledge of the labels associated with exemplars. This allows the method to be used also for tasks in which knowing the associated labels is not necessary, such as source separation.
- Using HACAM makes the splitting algorithm presented in Section III more efficient: As one of the child nodes *is* the parent node, for each active exemplar only one additional exemplar needs to be added.
- Since HACAM is medoid based, each parent node in the hierarchy is still an exemplar with its associated label. If instead the mean of two child nodes were used to represent the parent node, it is unclear what its associated label should be.
- The use of exemplars as medoids ensures no additional memory is required to store parent nodes.
- The use of medoids is more computationally efficient than standard HAC, as all distances only need to be calculated once. Accordingly, the computational complexity is $\mathcal{O}\left(J^2\right)$.
- The use of agglomerative clustering ensures that as you go down in the tree, the distances between the child nodes will decrease. Although there is no guarantee that added exemplars are beneficial to reducing the cost function, it ensures that adding new exemplars to the active dictionary has a smaller impact on the cost function in later iterations, which makes the algorithm more stable. Note that if we would have represented the parent node as the mean of the two child nodes, it can be easily shown that the cost function is non-increasing in the splitting procedure.
- Agglomerative clustering results in a *unique* hierarchy. Many divisive clustering approaches, on the other hand, yield hierarchies that depend on the initialisation in the k-means or k-median clustering used in each step.

V. Exemplar based noise robust ASR

As a test case for the proposed hierarchical exemplar-based sparse speech model, we use the obtained sparse representations for noise robust speech recognition as described in [8]. In this approach, we model noise spectrograms as a linear combination of noise exemplars $\mathbf{a}^{\mathrm{n}}_k$, with $k = 1, \ldots, K$ being the noise exemplar index. Representing the spectrogram of noisy speech by a vector $\mathbf{y}$, this leads to:

$$\mathbf{y} \approx \mathbf{s} + \mathbf{n} \tag{5}$$

$$\approx \sum_{j=1}^{J} \mathbf{a}^{\mathrm{s}}_j x^{\mathrm{s}}_j + \sum_{k=1}^{K} \mathbf{a}^{\mathrm{n}}_k x^{\mathrm{n}}_k \tag{6}$$

$$= [\mathbf{A}^{\mathrm{s}} \mathbf{A}^{\mathrm{n}}] \begin{bmatrix} \mathbf{x}^{\mathrm{s}} \\ \mathbf{x}^{\mathrm{n}} \end{bmatrix} \quad \text{s.t.} \quad \mathbf{x}^{\mathrm{s}}, \mathbf{x}^{\mathrm{n}} \geq 0 \tag{7}$$

$$= \mathbf{A}\mathbf{x} \quad \text{s.t.} \quad \mathbf{x} \geq 0 \tag{8}$$

with $\mathbf{x}^{\mathrm{n}}$ the sparse representation of noise. The matrix $\mathbf{A}$, the concatenated dictionary $[\mathbf{A}^{\mathrm{s}} \mathbf{A}^{\mathrm{n}}]$ with the speech and noise

exemplars as its columns, has dimensionality $E \times D$, where $D = J + K$. The sparse representation $\mathbf{x}$ is found by applying update rule (3) or (4), by substituting the speech-specific variables with their counterparts $\mathbf{x}$ and $\mathbf{A}$.

Each exemplar in the speech part of the dictionary $\mathbf{A}^{\mathrm{s}}$ is labelled using HMM-state labels obtained from a conventional MFCC-based decoder. Using a frame-by-frame state description of the training data used to construct the dictionary, we associate each exemplar $\mathbf{a}_j^{\mathrm{s}}$ with a label matrix $\mathcal{L}_j$, of dimensions $Q \times T$, with Q the total number of states in the system. The matrix $\mathcal{L}_j$ is a binary matrix containing for each frame $\tau \in [1, T]$ a single non-zero value for the corresponding active state. For each observed speech frame, we now calculate an unnormalised posterior probability vector as:

$$\mathbf{L} = \sum_{j=1}^{J} \mathcal{L}_j x_j^s \tag{9}$$

Subsequently, for each observed speech segment a state probability matrix is constructed by concatenating the unscaled posterior probability vectors. In order to decode utterances of arbitrary lengths, we adopt a sliding time window approach as in [8]. In this approach, we represent a noisy utterance as a number of fixed-size, overlapping speech segments. For each observed noisy speech segment, we calculate a state probability matrix as described above. Finally, we obtain a posterior probability matrix for the entire utterance by averaging the posterior probability estimates of the frames of all the windows that overlap, taking into account the exact temporal positions of the frames.

VI. Experiments

A. Experimental setup

For our experiments we used material from test set 'A' of the AURORA-2 corpus. Test set A comprises 1 clean and 24 noisy subsets, containing four noise types (subway, car, babble, exhibition hall) in six SNR bins. Each subset contains 1001 utterances with one to seven digits '0-9' or 'oh'. For testing we used the same random, representative subset of 10% of the utterances (i.e. 400 utterances per SNR level) used in [16].

In the experiments, the results of the four noise types are averaged. Acoustic feature vectors consisted of MEL frequency magnitude spectrograms, spanning $B = 23$ bands with a frame length of 25 ms and a frame shift of 10 ms. We used exemplars spanning 300 ms, thus $T = 30$ frames.

The noise dictionary consisted of $K = 4000$ exemplars randomly extracted from the noise sources underlying the noisy speech in the multi-condition training set of AURORA-2. The noise dictionary was kept fixed throughout all experiments: only the speech dictionary was hierarchically represented. The speech dictionary $\mathbf{A}^{\mathrm{s}}$ was constructed by sliding a window over the speech spectrograms in the clean training set in AURORA-2 and extracting every third exemplar (window position). The speech dictionary had size $J = 408\,066$. Prior to clustering with HACAM, the log was taken of the magnitude features and the dimensionality reduced to 100 by doing multiplication with a random matrix of dimensions 690×100. HACAM employed the Euclidean distance.

TABLE I: Computational effort in seconds spent on finding a single sparse representation, for various speech dictionary sizes. Note that the complete (speech + noise) dictionary size is $J + 4000$.

hierarchical dictionary	Speech dictionary size J					
	2000	4000	8000	16000	32000	64000
6.87	1.76	2.30	3.36	5.55	9.92	18.40

In our experiments with the hierarchical speech dictionary, we used an initial speech dictionary of $J_0 = 2000$ exemplars, with the exemplars determined by taking the top J_0 exemplars as determined by the hierarchy. We applied 600 iterations of the multiplicative update rule (4), and used a sparsity value of $\lambda = 0.65$ for speech exemplars and $\lambda = 0$ for noise exemplars (cf. [8]). In addition to the experiments with the splitting multiplicative update rule (4), we did experiments in which the dictionary was kept fixed (using update rule (3)). For these experiments we used several 'initial' dictionary sizes, $J_0 \in \{2000, 4000, 8000, 16000, 32000, 64000\}$,

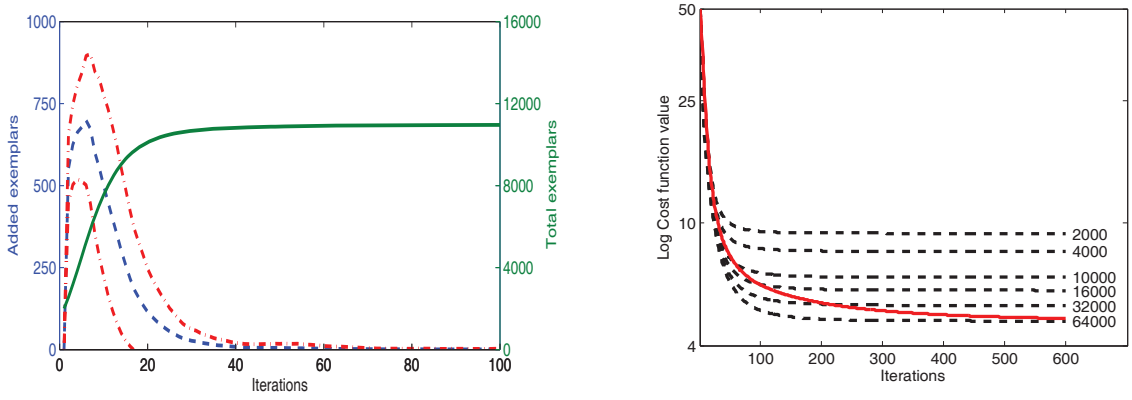
All algorithms were implemented in MATLAB, and utilized a GPU for acceleration using the Jacket [17] toolbox. The splitting multiplicative update algorithm was made efficient by using lookup tables which list the parent and child nodes of each exemplar at each point in the hierarchy. The machine used for the experiments was a Quad core Q6600 2.4 GHz, with 8 GB of RAM. It was equipped with a 1GB GTX460 GPU and operated under Windows 7, 64-bit with Matlab2010b. Implementation details of the speech decoding system can be found in [8]; In brief, digits were described by 16 states with an additional 3-state silence word, resulting in a 179 dimensional state-space. Each frame in each exemplars was labelled with an HMM-state as determined by forced alignment with a conventional HMM/MFCC-based recognizer.

In the experiments, only two SNR values, -5 dB as well as clean speech, are shown for brevity and since the focus of this paper is on the influence of the dictionary. For the same reason, we restrict ourselves to results on test set A, although this represents a best-case scenario as there is an overlap between the noise types in the noise dictionary and the noise types in the noisy speech. Experimental results in [8], [18], however, have also demonstrated good results on mismatched noise types.

B. Number of added exemplars and value of the cost function

To illustrate some properties of the proposed hierarchical approach, we gather statistics of finding a sparse representation on the single clean speech utterance "FCT_OOO9989A", which is represented by 248 sparse representations (corresponding to the positions of the sliding window). In Figure 2a we show the average number of exemplars that are added in each iteration. We only show the first 100 iterations, as after that only in some sparse representations exemplars were added in later iterations (up 25 at once, and up to 140 in total over the remaining 500 iterations). We can observe in Figure 2a that the average number of exemplars in this utterance is ≈ 11000. Incidentally, this average is representative for all utterances in the testset.

We can observe that most exemplars get added in the first ≈ 30 iterations, and almost all exemplars get added the first



(a) Exemplar counts

(b) Cost function

Fig. 2: Results of running the proposed algorithm for 600 iterations on a single utterance containing clean speech (258 sparse representations). Figure (a) shows the average number of speech exemplars added in each iteration (dashed lined), as well as the added number of speech exemplars plus and minus one standard deviation (dash-dotted lines). Also, the total number of speech exemplars in the dictionary is shown (solid line). Only the first 100 iterations are shown, as hardly any exemplars were added after that. Figure (b) shows the average value of the cost function (2) in each iteration, for the proposed method (solid line) as well various fixed dictionary sizes (dashed lines). The numbers right of the lines indicate the speech dictionary size for fixed dictionaries. Note that the y-axis is displayed on a log scale.

100 iterations. This is due to two effects: many exemplars have a positive weight change between iterations in the first few iterations (the solution is not very sparse yet) and the complete cluster tree is exhausted after only a limited number of levels. Investigation of the cluster structure reveals that even the deepest tree has only 113 branches. As stated above, it can still happen in some cases that exemplars can become active even after many iterations, but this is a rare occurrence.

In Figure 2b can observe the average value of the cost function (2) in each iteration. We have taken the average over all feature dimensions and all sparse representations in the utterance. We can observe the proposed algorithm starts out with a much larger value of the cost function than most dictionary sizes (equal, in fact, to $J_0 = 2000$, but converges to lower values, approaching the function value of $J_0 = 64000$ after 600 iterations. The fact that the value of the cost function shows a stable decrease (even when inspecting individual sparse representations rather than averages), shows that the added exemplars do, in fact, contribute to the reduction of the function value. After all, if arbitrary exemplars were added with the weight $0.5 * x^s$, it is unlikely the cost function value would decrease.

Finally, it we can observe that whereas the function values of the fixed dictionaries have pretty much converged after 300 iterations, the proposed method converges slower. This must be due to the fact that, as noted above, even after more than 300 iterations exemplars still get added to the dictionary occasionally. Incidentally, it should be noted that although the cost function may seem to have converged, the solutions may still become sparser with more iterations - in earlier work this was shown to improve performance [18].

C. Computational Effort

To roughly characterise the computational effort required by the proposed algorithm, we investigate running times for finding sparse representations. The running time for a fixed dictionary is completely determined by the dictionary size, feature dimension and the number of iterations. When using a hierarchical dictionary, there is an additional cost involved for bookkeeping and adding new exemplars. In Table I, we display the average time for finding a single sparse representation, averaged over 10 repeated runs using the same utterance used in Section VI-B. In this timing experiment, we limit the number of iterations to 150 as after that, only few exemplars get added when using the hierarchical dictionary approach.

In Table I we can observe that the running time of the proposed algorithm is between the running times of fixed dictionaries $J_0 = 16000$ and $J_0 = 32000$. Since we observed in Figure 2a that the average number of exemplars in this utterance is ≈ 11000, we can estimate that the running time almost doubled the first 150 iterations when compared to using a fixed dictionary. Profiling showed that most of the additional cost was due to finding which exemplars were active, rather than the process of adding the new exemplars. It is likely that finding of active exemplars can be further optimized, for example by not considering exemplars that are already a leaf node. Off course, regardless of such optimisations, the impact of adding the extra exemplars reduces as more iterations are used.

Finally, it should be noted that in modern computer architectures it is more efficient to find multiple sparse representations at once using matrix operations. This would require some modification of the proposed hierarchical approach, however, as in these approaches a single dictionary is shared between all sparse representations. Accordingly, one should split exemplars whenever a exemplar becomes active in *any* sparse representations. Implementation of such an approach is left as future work.

D. Speech recognition accuracy

The exemplar-based noise robust ASR system described in [16] relies on finding a linear combination of speech and

TABLE II: Recognition accuracies for various dictionary sizes.

SNR [dB]	hierarchical dictionary	Fixed speech dictionary size J					
		2000	4000	8000	16000	32000	64000
clean	97.0	77.9	83.7	89.9	92.3	95.0	96.3
-5	53.4	43.5	48.9	52.6	53.9	54.8	57.9

noise exemplars to provide noise robustness. While achieving impressive recognition accuracies on noisy speech, the method suffered on clean speech in comparison to conventional ASR methods. It was suggested that this could be due to, among other things, the fact only a few thousand speech exemplars were used.

In this experiment we explore the recognition on clean and noisy speech with the proposed approach, as well as with the fixed speech dictionaries used in the previous experiments. In Table II we can observe that the use of the hierarchical dictionary results in a clean speech accuracy of 97.0%, which is higher than the performance of $J_0 = 64000$, with accuracy of 96.3%. On noisy speech, the proposed method achieves 53.4% at SNR -5 dB., which situates it between the fixed dictionary sizes $J_0 = 8000$ and $J_0 = 16000$.

From the recognition results we can conclude that the proposed method is promising, as it achieves recognition accuracies that are higher than much larger (fixed) dictionaries at a much lower computational cost. In fact, an additional experiment (not shown) showed that this is a performance comparable to what is achieved with a dictionary of size $J_0 = 128000$, which results in 97.1% accuracy. At the same time, it is somewhat disappointing that the proposed method does not manage to perform better in noisy speech. However, there is likely to be a subtle interplay between the used value of the sparsity constraint and the number of added exemplars, an aspect that has not been explored. Also, the size of the initial dictionary, $J_0 = 2000$, is not a variable that has been explored exhaustively.

VII. Conclusions and future work

Based on the experimental results, we conclude that the proposed hierarchical exemplar-based speech model is promising starting point for future research. On clean speech, the method delivered a higher performance than the largest (fixed) exemplar dictionaries we tested, and also yielded a reduction in running time compared to large dictionaries.

Moreover, many avenues exist for future improvements. A very straightforward method would be to further increase the number of iterations, although this would also mean a relative increase in computational cost as when using fixed dictionaries, a lower number of iterations suffices. A more promising approach might be to add multiple exemplars at once for each active exemplar. This can be achieved for example by using ternary hierarchies or by descending into the hierarchy multiple levels at once. The corresponding accelerated growth of the dictionary could be reduced in several ways. First, there are the merging and splitting operations suggested in Section III-C. Also, the initial growth of the active dictionary can be reduced by adopting a 'burn-in' period in which no exemplars are added, or by modifying the criterion of which exemplars are active by only selecting exemplars that have a relative large weight as well as having an increasing weight.

Future work will consist of implementing the suggestions listed above, as well as exploration of different hierarchical clustering schemes.

Acknowledgment

The research of Jort F. Gemmeke was funded by IWT-SBO project ALADIN contract 100049.

References

[1] H. Bourlard, H. Hermansky, and N. Morgan, "Towards increasing speech recognition error rates," *Speech Communication*, vol. 18, pp. 205–231, 1996.

[2] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernolle, "Template based continuous speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1377–1390, 2007.

[3] H. Sakoe and S. Chiba, "A dynamic programming approach to continuous speech recognition," in *7th ICA*, 1971, p. 20 C13.

[4] C. Myers and L. Rabiner, "A level building dynamic time warping algorithm for connected word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 2, pp. 284 – 297, 1981.

[5] T. Deselaers, G. Heigold, and H. Ney, "Speech recognition with state-based nearest neighbour classifiers," in *Proc. INTERSPEECH*, 2007, pp. 2093–2096.

[6] P. Clarkson and P. J. Moreno, "On the use of support vector machines for phonemic classification," in *Proc. ICASSP*, 1999, pp. 585–588.

[7] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, February 2009.

[8] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," *accepted for publication in IEEE Transactions on Audio, Speech and Language processing*, 2011.

[9] J. F. Gemmeke, L. ten Bosch, L.Boves, and B. Cranen, "Using sparse representations for exemplar based continuous digit recognition," in *Proc. EUSIPCO*, Glasgow, Scotland, August 24–28 2009, pp. 1755–1759.

[10] T. N. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran, "Bayesian compressive sensing for phonetic classification," in *Proc. ICASSP*, 2010, pp. 4370–4373.

[11] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, April 2001, pp. 556–562.

[12] F. Sha, Y. Lin, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming," *Neural Computation*, vol. 19, pp. 2004–2031, 2007.

[13] J. F. Gemmeke, H. Van hamme, B. Cranen, and L. Boves, "Compressive sensing for missing data imputation in noise robust speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 272–287, 2010.

[14] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, "Proximal methods for sparse hierarchical dictionary learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 487–494.

[15] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*. Wiley, 1990.

[16] J. F. Gemmeke and T. Virtanen, "Noise robust exemplar-based connected digit recognition," in *Proc. ICASSP*, 2010.

[17] "Accelereyes Jacket," *Online: http://www.accelereyes.com/*, 2010.

[18] J. F. Gemmeke, A. Hurmalainen, T. Virtanen, and Y. Sun, "Toward a practical implementation of exemplar-based noise robust ASR," in *Proc. EUSIPCO*, 2011, pp. 1490–1494.