

# USING SPARSE REPRESENTATIONS FOR EXEMPLAR BASED CONTINUOUS DIGIT RECOGNITION

*Gemmeke et. al.*

Dept. of Linguistics, Radboud University  
P.O. Box 9103, NL-6500 HD  
Nijmegen, The Netherlands  
email: J.Gemmeke@let.ru.nl

## ABSTRACT

We have extended our previous work on isolated digit recognition by applying Sparse Classification to continuous digit recognition. The non-parametric technique is based on the idea that arbitrary speech signals can be represented as a linear combination of suitably selected exemplars. The classification is based on finding the smallest number of labeled exemplars in a very large library of exemplars that jointly approximate the observed speech token. We applied a sliding time window approach by applying SC to every window individually and decoding the utterance using Viterbi decoding. We show that our method outperforms KNN.

## 1. INTRODUCTION

For the last 30 years Automatic Speech Recognition (ASR) has been completely dominated by pattern recognition techniques based on Hidden Markov Models [1]. From a conceptual point of view this implies the assumption that (almost) all relevant phenomena in speech can be described in terms of a sequence of probabilistic models. This corresponds to the dominant position in Psycholinguistics that speech can be represented in the form of a relatively small number of discrete units (for example phonemes) that in one way or another abstract from the idiosyncrasies of individual tokens and that can be concatenated to create larger units such as syllables and words. More recently, a competing Psycholinguistic theory has been proposed, in which it is assumed that mental representations of speech include a record of detail of actual speech signals (called *episodes* or *exemplars*) that encode idiosyncrasies such as the speaker and possibly even the context in which an utterance was produced [2]. Interestingly, the episodic representations of speech proposed by that theory are reminiscent of the templates that formed the basis for the Dynamic Programming (or Dynamic Time Warping [DTW]) approach to speech recognition [3] that was superseded by HMMs in the late seventies of the previous century.

Compared to the DTW approach, HMMs had several decisive advantages: models that combined means with variances replaced templates, allowing more powerful distance measures and Viterbi decoding facilitated integrated search. HMMs were not only superior to DTW in conceptual terms, they also were better adapted to the limitations in memory and compute power of the digital computers and the algorithms that were available.

Recent advances in compute power, and the development of algorithms that can find structure in extremely large repositories of observations, may make episodic approaches computationally feasible. At the same time it has become clear that not all speech phenomena can be covered in the form of HMMs. There is general agreement in the speech community about the need for novel approaches, not to fully replace HMMs, but certainly as an addition for handling phenomena that HMMs do not account for [1, 4]. Therefore, it is interesting to revisit pattern matching techniques such as DTW and episodic representations, to investigate if these can handle problems that are notoriously difficult to solve in an HMM framework. One such problem is speech recognition in adverse acoustic conditions.

In [5] it was shown that a template-based approach of isolated digit recognition in noise can outperform conventional model-based approaches. The approach, dubbed *sparse classification (SC)*, is based on the idea that all speech signals can be represented as a linear combination of suitably selected exemplars. The classification is based on finding the smallest number of labeled exemplars in a very large library of exemplars that jointly approximate the observed speech token. Because there is no need for these exemplars to be close to each other in the original space, Sparse Classification differs from the usual interpretation of episodic recognition and other exemplar-based approaches to speech recognition, based on searching the exemplars with the smallest distance to the observed speech token.

As a step toward noise robust continuous speech recognition we extend our previous work on isolated digit recognition by applying Sparse Classification to continuous digit recognition. To keep the enterprise manageable we will apply SC to noise-free continuous digit sequences, leaving the extension to noise robustness for future research. However, we will compare SC to a K-nearest-neighbor (KNN) approach to finding exemplars in a large library. By doing so we investigate differences and similarities between sparse classification and KNN in more detail.

## 2. METHOD

### 2.1 Exemplar-based classification

In ASR speech signals are represented as a spectro-temporal distribution of acoustic power, called a spectrogram, which in its turn is represented as a  $B \times T$  dimensional matrix (with  $B$  frequency bands and  $T$  time frames).

We express the spectrogram  $S$  as a single vector  $s$  of dimension  $D = K \cdot T$  by concatenating  $T$  subsequent time frames. We have a matrix  $A$  of exemplar spectrograms, obtained from a training database, formed as  $A = (\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_{N-1} \ \mathbf{a}_N)$  with  $\mathbf{a}_n$ , ( $1 \leq n \leq N$ ) a specific token in the set of  $N$  available exemplars. With  $\mathbf{a}_n$  reshaped from a spectrogram just like  $s$ , the matrix  $A$  has dimensionality  $D \times N$ . Both  $s$  and the columns of  $A$  are normalized to unit (Euclidean) norm.

For speech recognition the reshaped exemplar spectrograms  $\mathbf{a}_n$  must have labels corresponding to linguistic classes (words, syllables, phonemes, etc.). Since  $\mathbf{a}_n$  typically contains multiple time frames, an exemplar may be associated to more than one class. For example, if the exemplar contains exactly one word, the exemplar is associated not only with that particular word, but also with all syllables or phonemes in the word. We map every exemplar  $\mathbf{a}_n$  to a label vector  $\mathbf{y}_n$ . Denoting the total number of possible classes with  $Q$ ,  $\mathbf{y}_n$  is a binary vector of length  $Q$  of which the nonzero elements indicate with which classes  $\mathbf{a}_n$  is associated.

We obtain a label matrix  $Y$  of dimensions  $Q \times N$  by concatenating all exemplar labels  $\mathbf{y}_n$ :  $Y = (\mathbf{y}_1 \ \mathbf{y}_2 \ \dots \ \mathbf{y}_{N-1} \ \mathbf{y}_N)$ . Finally we denote the unknown label vector associated with the observed speech  $s$  as  $\mathbf{y}^s$ .

For recognition we first obtain a vector  $\mathbf{x}$  of length  $N$  that maps the observed speech vector  $s$  to exemplars in  $A$ . The weight vector

$\mathbf{x}$  tells us which exemplars are associated with the observed speech. We then obtain the label vector  $\mathbf{y}^s$  by calculating the score  $\mathbf{f}_s$  as:

$$\mathbf{f}_s = Y\mathbf{x} \quad (1)$$

with  $\mathbf{x}$  a sparse column vector and  $\mathbf{f}_s$  with length  $Q$ . Keeping track of only the nonzero elements of  $\mathbf{f}_s$  we obtain the binary label vector  $\mathbf{y}^s$ . In this paper we compare KNN and SC approaches to find a small number of examples that best represent  $\mathbf{y}$ .

## 2.2 Nearest Neighbor classification

In exemplar-based speech recognition we can use nearest neighbor (NN) techniques to associate an unknown speech token  $S$  with exemplars from a dictionary  $A$  which are closest to  $S$  given some distance measure. For every exemplar  $\mathbf{a}_n$  we obtain the Euclidean distance to  $s$  and we retain the  $K$  exemplars with the smallest distance. We then create a binary vector  $\mathbf{x}$  of length  $N$  with  $K$  nonzero elements pertaining to the indices of the exemplars with the smallest distance.

## 2.3 Sparse Classification

In our sparse classification approach we try to find exemplars which *jointly* approximate the observed speech token. As in [5], we assume we can represent  $s$  as a linear combination of the exemplar spectrograms  $\mathbf{a}_n$ . We write:

$$\mathbf{s} = \sum_{n=1}^{N_A} x_n \mathbf{a}_n = A\mathbf{x} \quad (2)$$

Depending on the dimensionality  $D$  and  $N$ , this system of equations can be *over* or *under*-determined. Research in the field of compressive sensing [6, 7] and error-correction [8] has shown that if  $\mathbf{x}$  is *sparse*, in both regimes it can be recovered by finding the *sparsest* solution. With mild conditions on the sparsity of  $\mathbf{x}$  and the structure of  $A$ ,  $\mathbf{x}$  can also be recovered using  $l^1$  minimization:

$$\min_{\mathbf{x}} \{ \|\mathbf{x}\|_1 \} \text{ subject to } \mathbf{s} = A\mathbf{x} \quad (3)$$

This convex minimization problem can be cast as a least squares problem with a sparsity enforcing  $l^1$  penalty:

$$\min_{\mathbf{x}} \{ \|A\mathbf{x} - \mathbf{s}\|_2 + \lambda \|\mathbf{x}\|_1 \} \quad (4)$$

with a regularization parameter  $\lambda$ . If  $\mathbf{x}$ , with sparsity  $f = \|\mathbf{x}\|_0$ , is very sparse, Eq. 4 can be solved efficiently in  $\mathcal{O}(f^3 + N_A)$  time using homotopy methods [9]. A benefit of this technique is that  $\lambda$  does not have to be determined in advance. **Zonder uitleg erbij zie ik de toevoegde waarde van de laatste zin niet.**

## 2.4 Classification of continuous speech

Classification as described above indirectly requires that all spectrograms  $S$  have a fixed time dimension  $T$ . In practice, utterances in continuous speech can be of arbitrary length. Classification by (automatically) segmenting the speech signal prior to decoding is error-prone and would require some form of subsequent time normalization. Instead, we will perform classification using a sliding time window of fixed length.

Consider the spectrographic representation  $U$  of a speech utterance of length  $I$  time-frames. We then slide a window  $S$  of length  $T$  through  $U$ , with increments of  $\Delta$  frames. If  $\Delta = 1$  (and  $T > 1$ ) we have maximally overlapping windows. If  $\Delta = T$  we do not have any overlap and if  $\Delta > T$  a number of frames in  $U$  will be entirely unseen. A visual representation is shown in Fig. 1.

The number of windows  $W$  needed for processing the entire speech signal  $U$  is given by  $W = (\text{ceil}((I - T)/\Delta)) + 1$ . Only in the last window care must be taken: If  $\Delta > 1$  it may happen that the number of frames  $T$  in the sliding window is larger than the number of frames left in the utterance. The number of frames in the

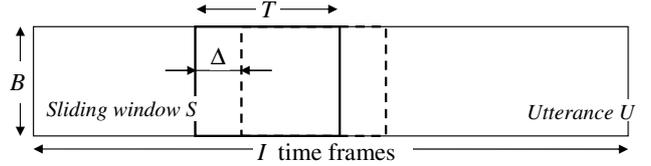


Figure 1: Schematic diagram of time-continuous classification using overlapping windows.

final window is given by  $Z = I - W \cdot \Delta$ . Classification in the final window is done by removing the bottom  $(T - Z) \cdot K$  rows from the dictionary after which classification proceeds as normal.

Applying the procedure described in Section 2.1 we obtain a score vector  $\mathbf{f}_w$  for every window  $w \in W$ . Rather than converting these to binary labels  $\mathbf{y}^w$  we use the scores directly. We concatenate all scores in a matrix  $F$  of size  $Q \times W$ . In this matrix, the classification result of the entire utterance can be found by finding a path through time which maximizes the score. An example of the score matrix is shown in Fig. 2.

## 2.5 Viterbi decoding

So far, we have made no assumptions about the kind of class labels we want to classify. The eventual goal of classification is a word-based transcription, but that leaves room for smaller classification units. In conventional HMM-based speech recognition the underlying classification units are *states*, and every word or phoneme is composed of sequences of such states. In that case, the search for the best state sequence through the Trellis matrix  $F$  is constrained by the state sequences in the acoustic word models. In our current example-based approach, however, we are free to associate exemplars with either states, phonemes, entire words or any other unit.

In this paper, we consider two different representations: a state-based representation and a word-based representation. The choice of this unit implies different constraints on the decoding strategy. In case the Trellis matrix  $F$  is build by using word labels, for example, the optimal path through  $F$  must avoid a too frequent hopping between word hypotheses which would lead to too many insertions of too short words. Usually these insertions can be reduced by adjustment of the word entrance penalty, but it is well known that this does not necessarily prevent the emergence of evidently too-short words along the best path. In this subsection, we will discuss the approach that was followed to impose duration constraints.

### 2.5.1 State-based labelling

In the case of the *state-labelled* decoding, we used the conventional viterbi algorithm that was available as back-end of an HMM-based speech decoder described in [10]. In this case, state-hopping is constrained by the state-sequence structure in the acoustic word models, while word-word transitions are modulated by the word entrance penalties (and the word-based LM, if any).

### 2.5.2 Word-based labelling

In the case of *word-labelled* decoding we applied a dedicated viterbi algorithm that has been modified to be able to incorporate minimum and maximum duration constraints on the lengths of same-state sequences along the best path. This modified algorithm is equivalent to the standard Viterbi algorithm for finding the best path through a (rectangular) Trellis local cost matrix  $F$ , with two modifications. These modifications are described in detail below.

We assume that a rectangular matrix  $F$  is specified with non-negative costs. In the previous section, its column index and row index corresponded to the window index  $w$  and class index  $q$ , respectively. However, since the (modified) Viterbi algorithm operates on any kind of rectangular matrix, in this subsection we will

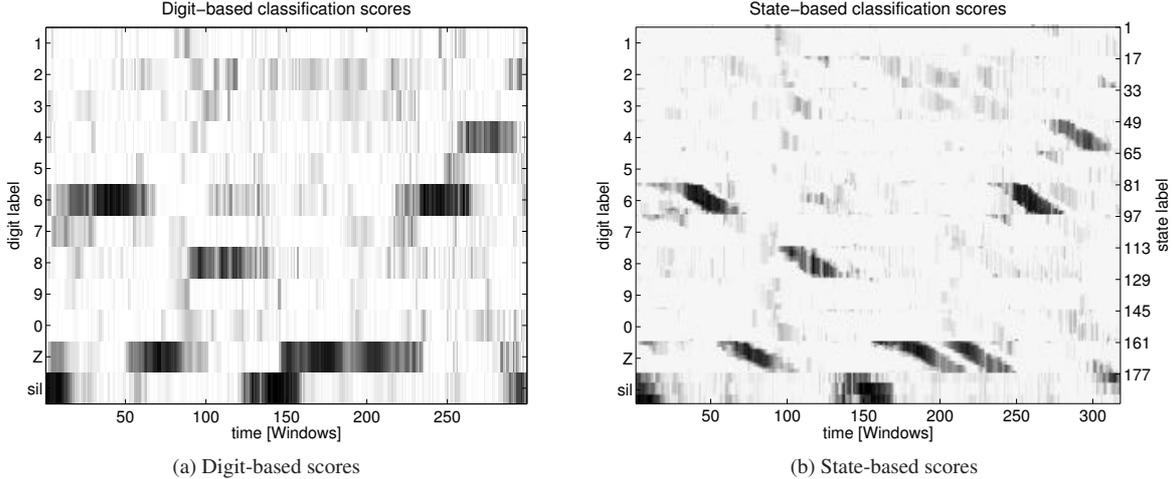


Figure 2: Example of time-continuous classification scores obtained using sparse classification, a window length of 20 frames and a dictionary size of 16000 exemplars. Fig. 2a shows scores obtained using digit-based class labels and the scores displayed in Fig. 2b represent HMM-state based labels. The corresponding digit labels are also shown. The utterance spoken is ‘6Z8ZZ64’, with ‘Z’ representing ‘zero’. The label ‘sil’ represents silence.

use the default notation of  $i$  and  $j$  referring to indicate the column and row indices. The size of  $F$  is  $W$  (columns) by (vertical)  $Q$  (rows).

The aim is to find the best (cheapest) path starting at *any* point at the left side ( $\{(1, j), 1 \leq j \leq Q\}$ ) of the matrix to *any* point at the right side ( $\{(W, j), 1 \leq j \leq Q\}$ ) of the matrix, in such a way that the horizontal stretches of this path (i.e. parts of the path with same  $j$ ) obey minimum and maximum length constraints. To that end, for each value of  $j$  the minimum and maximum duration is specified by means of two user-specified arrays  $\min(j)$  and  $\max(j)$  ( $1 \leq j \leq Q$ ).

The two modifications are as follows.

- The set of predecessors. In the classical Viterbi implementation, usually the best (cheapest) path is sought between the lower left point  $(1, 1)$  and upper right point  $(W, Q)$  of the Trellis matrix  $F$  allowing only transitions from neighboring points  $\{(i-1, j), (i-1, j-1), (i, j-1)\}$ . In the modified Viterbi algorithm, however, we allow the set of predecessors of the point  $(i, j)$  to the set  $\{(i-1, k)\}_{1 \leq k \leq Q}$ .
- Transition penalties. The second modification deals with the penalties that are associated with each transition step, for two reasons (a) the set of predecessors is larger than in conventional viterbi and (b) a penalty  $\alpha$  must be imposed when duration constraints are violated.

As can be inferred from Algorithm 1, transition penalties were implemented as follows. Given the matrix of local costs  $F$ , we define two auxiliary matrices of the same size as  $F$ . The first auxiliary matrix (which is also applicable in conventional Viterbi) is a matrix  $G$  that will eventually contain the *global* costs. The second auxiliary matrix is  $D$ , that will be used during the search such that the integer  $D(i, j)$  eventually specifies the duration of the most recent word hypothesis along the best path from the left-hand side to  $(i, j)$ . Furthermore, we already have the two arrays  $\min(j)$  and  $\max(j)$  that specify minimum and maximum durations for each  $j$ ,  $1 \leq j \leq Q$ . And possibly there is a language model  $LM(q_i, q_j)$  that specifies the cost of the bigram  $q_i \rightarrow q_j$  for all class pairs  $q_i, q_j$ . **LTB: notatie overbelast hier**

After the recursion, the backtrace based on the values of  $k_{min}$  for  $i = W, W-1, \dots, 1$  provides the cheapest path through the Trellis matrix  $F$  taking into account the imposed duration constraints.

The term  $LM$  specifies the language model costs of having a transition between one class to another class. The balancing between the local costs in the  $F$  matrix, the penalties in  $LM$ , and  $\alpha$  determines the behavior of the algorithm in terms of the best path. The

#### Initialisation step:

```

for  $j = 1$  to  $Q$  do
   $G(1, j) = F(1, j); D(1, j) = 1;$ 
end for

```

All other entries in  $F$  and  $D$  will be defined later in the search, but could be initialized to 0

#### Recursion step:

```

for  $i = 2$  to  $W$  do
  for  $j = 1$  to  $Q$  do
    •  $G(i, j) = \min_{k \in \{1 \dots Q\}} \{G(i-1, k) + LM(k, j) + C_{dur} + F(i, j)\}$ 
    •  $k_{min} =$  the minimizing value for  $k$ 
    •  $D(i, j) = \begin{cases} D(i-1, k_{min}) + 1 & \text{if } k_{min} = j \\ 1 & \text{otherwise} \end{cases}$ 
  end for
end for

```

In this scheme,  $C_{dur}$  denotes the duration violation cost which is dependent on  $i, j, k, D(i-1, k), \min(k)$ , and  $\max(k)$  and is specified as:

$$C_{dur} = \begin{cases} \alpha & \text{if } k \neq j \text{ and not } \min(k) \leq D(i-1, k) \leq \max(k) \\ 0 & \text{if } k \neq j \text{ and } \min(k) \leq D(i-1, k) \leq \max(k) \\ \alpha & \text{if } k = j \text{ and } D(i-1, k) > \max(k) \\ 0 & \text{if } k = j \text{ and } D(i-1, k) \leq \max(k) \end{cases}$$

#### Algorithm 1: Modified Viterbi algorithm

higher the value of  $\alpha$ , the less favorable it becomes to change states while violating the duration constraints – in the limit ( $\alpha \rightarrow \infty$ ), only durations of class  $k$  are only allowed between  $\min(k)$  and  $\max(k)$ .

## 3. EXPERIMENTS

### 3.1 Experimental setup

The speech material used for evaluation is the clean speech test set ‘A’ of the AURORA-2 corpus [11]. This corresponds to a subset of the TIDIGIT corpus [12]. Acoustic feature vectors consisted of mel frequency log power spectra:  $K = 23$  bands with center frequencies starting at 100 Hz.

Window [T]	Dictionary size [N]						
	250	500	1000	2000	4000	8000	16000
1	5.0	10.5	22.7	31.2	38.1	43.8	49.2
5	20.9	32.9	41.0	48.5	55.3	62.1	65.1
10	36.8	44.1	57.9	63.4	68.1	70.6	76.4
20	46.4	55.5	67.4	72.7	78.0	80.7	84.7
35	49.4	63.2	71.3	77.9	82.2	84.4	85.8
50	41.9	52.1	58.4	65.5	69.4	71.4	69.9

Table 1: Recognition results for several window lengths and dictionary sizes. The results shown here pertain to SC classification with word-based decoding.

The dictionary was formed by exemplar vectors, which are reshaped versions of spectrograms (each spanning  $T$  frames). In contrast to the isolated digit approach described in [5], however, the exemplar spectrograms in the dictionary are now created by extracting spectrogram fragments with a random offset from randomly selected utterances in the train set of AURORA-2. For every window length  $W$  we created a dictionary with 16000 spectrogram fragments of length  $T$ . These fragments were extracted from randomly selected utterances with a random offset. The fragments were reshaped to vectors and subsequently added as the columns of the dictionary. For experiments with dictionary sizes smaller than 16000 we used the first  $N$  columns of the original dictionary to form the final dictionary  $A$ .

HMM-state based labels of the train set were obtained via a forced alignment with the orthographic transcription using a conventional HMM-based speech decoder. Digits were described by 16 states with an additional 3-state silence word. From the frame-by-frame HMM-state labels we could extract higher level labels such as digit labels. Minimum and maximum digit durations were also extracted from the state-based transcription.

The speech decoding system was implemented in MATLAB. The  $l^1$  minimization for sparse classification was carried out using the `SolveLasso` solver implemented as part of the `SparseLab` toolbox which can be obtained from [www.sparselab.stanford.edu](http://www.sparselab.stanford.edu). This iterative method was terminated after 30 iterations, resulting in (at most)  $f = 30$  nonzero coefficients. Correspondingly we use the  $z = 30$  nearest neighbors when doing KNN classification.

### 3.2 Window length and dictionary size

Sparse Classification introduces several new parameters, such as the number of clean exemplars and the duration (number of frames) of these exemplars. For continuous speech at least one additional parameter is introduced, viz. the step size with which exemplars are matched against the signal to be processed. Larger step sizes reduce computational effort but can decrease recognition accuracy. Because the parameters might well show significant interactions, this paper will investigate the recognition accuracy as a function of the number of exemplars and the duration while keeping the step size constant at  $\Delta = 1$  frame.

We carried out recognition experiments with a number of dictionary sizes  $N$  and window length  $T$ . We consider window lengths of 1, 5, 10, 20, 35 and 50 frames and dictionary sizes of 250, 500, 1000, 2000, 4000, 8000, 16000 exemplars. We used two decoding schemes: HMM-state-based labeling and word-based labeling. For every window, we first determine the associated exemplars, both through KNN and SC classification. For both methods we then do decoding twice: once using word-based labels (Tables 1 and 3) and once using the low level HMM-state labeling (Tables 2 and 4).

## 4. RESULTS AND DISCUSSION

### 4.1 word-labeling vs. state-labeling

When comparing word-based decoding results shown in Tables 1 and 3 with the HMM-state-based results in Tables 2 and 4 we

can observe that HMM-state-based decoding delivers much better recognition accuracy. This difference can be observed both when looking at the maximum recognition accuracy (98.2% vs 85.5% for SC and 94.5% vs 75.1% for KNN) as well for the lowest recognition accuracies reported (16.8% vs 8.6% for SC and 10.1% vs 3.2% for KNN).

For both decoding approaches the best recognition accuracies are obtained with a dictionary size of 16000 exemplars. The optimum with respect to window is different: For word-based decoding the best recognition accuracies are obtained with a window length of 20 (SC) or even 35 (KNN) frames. For HMM-state-based decoding the best accuracies are obtained with a window length of 10 frames.

A detailed analysis (not shown here) showed that we make many more insertion and (to a lesser extent) deletion errors when doing word-based decoding. The main reason is that during word-based decoding we have no way to distinguish between the begin and the end of a word. The HMM-state models, on the other hand, are very structured due to their 16-state state sequence. As can be observed in Fig. 2a any spurious digit activation which adheres to the (necessarily small) minimum duration may cause an insertion error. In Fig. 2a we can observe that spurious state activations do not lead to insertions due because the Viterbi search is required to find the clear diagonal structures that constitutes the state sequence underlying digits. This also explains the difference in optimum window length: larger window lengths will cause less short-duration spurious digit activations thus reducing insertions.

Similarly, deletions can occur when encountering repeated words. As can be observed in the example in Fig. 2 the repeated digit '6' will cause a deletion if the combined length of the two digits does not exceed the maximum duration for this digit. When doing HMM-based state decoding we can clearly distinguish two separate digits.

The downside of HMM-state-based decoding is that we need a frame-by-frame state transcriptions of the train set, obtained with a HMM-based recognizer. Instead it is probably advantageous to move to a phone-based representation is in [4]. This would also give robustness against errors like repeated digits and insertions and additionally naturally extends to larger vocabulary tasks.

### 4.2 Window length

When comparing the results with respect to window lengths, we notice that SC totally fails when using only a single frame. KNN classification, on the other hand, performs very good using a window that spans a single frame. Both classification methods show decreasing recognition accuracies at large window sizes.

**Don't move this to the margin:placeholder Turns out the non-negativity constraint actually does make a difference for small dict and/or window sizes. The reason is that the decoding cannot (at the moment) handle negative values of  $\text{vec}(x)$ . Conceptually speaking I have no clue what negative weights on exemplars should mean anyway...Unfortunately, the used solver has a bug and thus not, in fact, give non-negative results. Sigh. It turns out, however, that simply taking the  $\text{abs}()$  of the, possibly negative result, already improves things substantially...**

Window [T]	Dictionary size [N]						
	250	500	1000	2000	4000	8000	16000
1	36.4	40.3	50.8	61.0	69.0	76.3	80.4
5	64.7	78.7	87.8	93.1	95.6	96.9	97.5
10	65.8	82.8	92.1	95.3	96.9	97.4	98.0
20	69.1	85.0	92.4	94.0	95.6	96.1	96.7
35	64.5	73.9	79.8	83.1	84.8	85.1	83.6
50	48.0	58.7	64.1	68.1	69.7	67.9	63.8

Table 2: Recognition results for several window lengths and dictionary sizes. The results shown here pertain to SC classification with HMM-state based decoding.

Window [T]	Dictionary size [N]						
	250	500	1000	2000	4000	8000	16000
1	9.0	17.1	25.7	32.3	35.6	39.5	38.6
5	8.5	16.7	16.4	13.2	21.1	28.9	29.0
10	4.9	18.2	41.9	44.3	41.2	43.0	46.0
20	2.7	5.6	33.6	44.8	56.5	62.0	64.9
35	7.5	17.1	31.0	50.2	61.5	69.8	75.1
50	3.2	14.9	27.1	37.8	51.0	59.4	65.9

Table 3: Recognition results for several window lengths and dictionary sizes. The results shown here pertain to KNN classification with word-based decoding.

Window [T]	Dictionary size [N]						
	250	500	1000	2000	4000	8000	16000
1	26.3	43.6	60.4	73.1	84.9	89.1	91.9
5	29.1	62.1	83.9	91.2	92.4	92.9	93.4
10	21.1	44.9	79.0	89.8	93.1	93.9	94.5
20	12.6	31.9	64.3	81.5	89.1	91.3	92.9
35	10.1	21.4	40.3	59.6	72.0	78.0	81.6
50	11.6	14.6	30.6	43.0	55.5	62.4	67.3

Table 4: Recognition results for several window lengths and dictionary sizes. The results shown here pertain to KNN classification with HMM-state based decoding.

The reduced accuracy at large window lengths may due to the fact we our exemplars now span entire digits, overlapping digits and occasionally even three digits per window. It is likely that we need (much) more exemplars to express the full range of possible digit-digit transitions, time-shifts & duration variations when using these large window sizes.

For HMM-state-based decoding we observed that the best accuracies are obtained with a window length of 10 frames. This is roughly the same number of frames which are taken into account when using delta and delta-delta features in traditional HMM-based decoding.

#### 4.3 Dictionary size

When comparing the results with respect to dictionary size it is clear that we obtain higher accuracies with larger dictionary sizes. The increase in recognition accuracy when increasing the dictionary sizes beyond 4000 exemplars is sub linear, however. This is most likely due to the fact that the additional exemplars are often very similar to exemplars already present in the dictionary and thus carry little, if any, additional information. While creating a dictionary from randomly selected exemplars was effective for dictionary sizes up to 4000 or 8000, it is probably more efficient to turn toward more deterministic approach to create the exemplar basis such as Vector Quantization (VQ).

Another difference between SC and KNN based classification becomes clear when comparing recognition at very small ( $\leq 1000$  exemplars) dictionary sizes. At these dictionary sizes SC performs much better. This is most likely due to the fact that no single exemplar really resembles the observed speech token, causing a bad choice using KNN. SC on the other hand can combine exemplars to jointly approximate the observed speech token. Another issue is that having to choose 30 nearest neighbors from such a small dictionary is bound to include many exemplars with labels that do not correspond to the observed speech token. SC on the other hand does not overfit: the algorithm has the freedom to use far less than 30 exemplars.

#### 4.4 Noise robust ASR

The good recognition accuracies reported for SC 98.2% not only are better than those obtained using KNN, but also make the way free to improve noise robustness using Missing Data Technique (MDT)

[13]. At the heart of MDT is the assumption that it is possible to estimate –prior to decoding– which spectro-temporal elements of the acoustic representations are reliable (i.e., dominated by speech) and which are unreliable (i.e., dominated by background noise). One way of noise robust speech decoding is to base recognition only on features which are labeled reliable.

The Compressive Sensing theory the SC method is based on asserts that a sparse representation of a signal can be recovered from a very limited number of measurements (features). In [5] we showed for isolated digits that SC can - provided a sufficiently accurate missing data mask is provided - can successfully recognize digits using very few reliable features. The good recognition accuracies obtained with SC on this connected digit task warrants further research in this direction.

## 5. CONCLUSIONS

We have extended our previous work on isolated digit recognition by applying Sparse Classification to continuous digit recognition. The non-parametric technique is based on the idea that arbitrary speech signals can be represented as a linear combination of suitably selected exemplars. The classification is based on finding the smallest number of labeled exemplars in a very large library of exemplars that jointly approximate the observed speech token. We applied a sliding time window approach by applying SC to every window individually and decoding the utterance using Viterbi decoding. The classification within a single window is based on finding the smallest number of labeled exemplars in a very large library of exemplars that jointly approximate the observed speech token. Our results show SC compares favorably compared to a KNN approach, achieving a recognition accuracy of 98.2% vs 94.5% for KNN.

We have compared word-based and HMM-state based decoding and showed that HMM-state-based recognition gives superior recognition accuracies because we can distinguish between the begin and end of words, thus preventing insertion errors from spurious word activations and preventing deletion errors in the presence of repeated digits.

We have also investigated the influence of window length and dictionary size and showed that the best recognition accuracies are obtained using as large a dictionary as possible and a window length of 10 frames for HMM-state-based decoding.

The promising results of SC opens a new direction of research in exemplar-based speech recognition and makes the way free for future research on the noise robustness properties reported on in [5].

## 6. ACKNOWLEDGMENTS

The research of Jort Gemmeke was carried out in the MIDAS project, granted under the Dutch-Flemish STEVIN program.

## REFERENCES

- [1] H. Bourlard, H. Hermansky, and N. Morgan, "Towards increasing speech recognition error rates," *Speech Communication*, vol. 18, p. 205231, 1996.
- [2] J. Goldinger, "Echoes of echoes? an episodic theory of lexical access," *Psychological Review*, vol. 105, pp. 251–279, 1998.
- [3] C. Myers and L. Rabiner, "Connected digit recognition using a level-building dtw algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, pp. 351 – 363, 1981.
- [4] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernelle, "Template based continuous speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1377–1390, 2007.
- [5] J. Gemmeke and B. Cranen, "Noise robust digit recognition using sparse representations," in *Proceedings of ISCA 2008*

*ITRW "Speech Analysis and Processing for knowledge discovery"*, 2008.

- [6] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [7] E. J. Candès, "Compressive sampling," in *Proc. of the International Congress of Mathematicians*, 2006.
- [8] E. J. Candès and R. P., "Highly robust error correction by convex programming," *IEEE Trans. Inform. Theory*, vol. 54, pp. 2829–2840, 2006.
- [9] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.
- [10] H. Van hamme, "Prospect features and their application to missing data techniques for robust speech recognition," in *Proc. of INTERSPEECH-2004*, 2004, pp. 101–104.
- [11] H. Hirsch and D. Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Proc. of ISCA ASR2000 Workshop, Paris, France*, 2000, pp. 181–188.
- [12] R. G. Leonard, "A database for speaker-independent digit recognition," in *Proceedings of ICASSP 84*, 1984, pp. 42.11.1 – 42.11.4.
- [13] B. Raj and R. M. Stern, "Missing-feature approaches in speech recognition," *Signal Processing Magazine*, vol. 22, no. 5, pp. 101–116, 2005.