# Improvements in Neural-Network Training and Search Techniques for Continuous Digit Recognition

*John-Paul Hosom[†], Ronald A. Cole[§], and Piero Cosi[‡]*

*[†]Center for Spoken Language Understanding (CSLU-OGI)*
*Oregon Graduate Institute (OGI), P.O. Box 91000, Portland, Oregon 97291 USA*
*email: hosom@cse.ogi.edu*
*[§]Center for Spoken Language Understanding (CSLU-Boulder)*
*University of Colorado, Boulder, Campus Box 258, Boulder, Colorado 80309 USA*
*email: cole@cslu.colorado.edu*
*[‡]Institute of Phonetics -- C. N. R., Via G. Anghinoni, 10 - 35121 Padova ITALY*
*email: cosi@csrf.pd.cnr.it*

***ABSTRACT*** *This paper describes a set of experiments on training and search techniques for development of a neural-network based continuous digits recognizer. When the best techniques from these experiments were combined to train a final recognizer, there was a 56% reduction in word-level error on the continuous digits recognition task. The best system had word accuracy of 97.67% on a test set of the OGI 30K Numbers corpus; this corpus contains naturally-produced continuous digit strings recorded over telephone channels. Experiments investigated the effects of the feature set, the amount of data used for training, the type of context-dependent categories to be recognized, the values for duration limits, and the type of grammar. The experiments indicate that the grammar and duration limits had a greater effect on recognition accuracy than the output categories, cepstral features, or a doubling of the amount of training data. In addition, the forward-backward method of training neural networks was employed in developing the final network.* ***Key words: speech recognition, neural networks, digit recognition.***

## 1. Introduction

Despite many improvements over the years in feature representation and training methods, the performance of automatic speech recognition systems is still far inferior to human performance, especially for telephone-channel speech. Our current research on improving recognition performance focuses on the telephone-speech continuous digits task. Despite its apparent simplicity, this task is known to be quite difficult, in large part because statistical language models can not be employed.

The platform for research and development of our recognizers is the CSLU Toolkit [1], described in more detail below. We have developed a set of procedures within the Toolkit for training recognizers on tasks such as continuous digit recognition, and a step-by-step tutorial is available [2]. The recognition systems described in this paper are frame-based hybrid HMM/ANN recognizers with context-dependent categories. The method for training such systems is simple enough that a bright high-school student can complete the tutorial in a few days. On the continuous digits task, the training procedure yields recognition results that compare favorably to standard HMM systems [3]. This paper shows how competitive performance was achieved by optimizing several of the parameters used in training and searching. Our experiments focused on the choice of feature set, the amount of data used for training, the type of context-dependent categories to be recognized, the values for duration limits, and the type of grammar[1].

## 2. The CSLU Toolkit

### 2.1. Toolkit Overview

The CSLU Toolkit is a comprehensive software environment that integrates a set of speech-related technologies, including speech recognition, natural-language parsing, speech synthesis, and facial animation. The Toolkit has been developed to support speech-related research and development activities for a wide range of users and uses, and it features GUI authoring and analysis tools that enable rapid development of desktop and telephone-based speech applications. It is currently used, among other places, at the Tucker-Maxon Oral School in Portland, Oregon, to create interactive learning experiences for both hearing and profoundly deaf children. The Toolkit is available at no charge for academic use, and can be downloaded over the Internet [4]. Among other topics, the Toolkit is designed to enable users to:

- rapidly design spoken language systems for real applications with easy-to-use authoring tools, even if the user is unfamiliar with spoken language technology;
- learn about spoken-dialogue systems as well as the fundamentals of speech through coursework incorporated into the tools; and
- perform research on the underlying technologies and incorporate research advances into working systems for evaluation in real applications.

[1] Although a grammar in which any word can follow any other word is quite simple, there are some implementation choices that, as will be seen, can have a large impact on recognition performance.

### 2.2. Levels of the Toolkit

#### *2.2.1 Toolkit Core Level*

The core of the Toolkit consists of a set of modules that implement technology that is fundamental to all aspects of speech processing and facial animation. These modules are written in C and form an application programming interface (API) that is hardware and operating-system independent. The modules contain routines for signal processing, training artificial neural networks (ANNs) and Hidden Markov Models (HMMs), pipelined speech recognition with a Viterbi search, and telephone and microphone interfaces. The modules also include a robust natural-language parser [5], an enhanced version of the Festival text-to-speech system [6] originally developed at the University of Edinburgh, an animated talking face called Baldi [7] that was developed at the University of California, Santa Cruz, and public-domain dictionaries. The modules are grouped into functional libraries that are dynamically linked and loaded at run time, allowing the application to scale in terms of resources. The modules can be linked directly into a C program or individually loaded into a programming shell, as needed. This architecture allows the Toolkit to be easily extended by an individual researcher. Great care was taken to design all of the core components to operate in as efficient and consistent a manner as possible, with special attention given to modularity, portability, and extendibility.

#### *2.2.2 Toolkit Shell Level*

The main application level of the Toolkit is a programming shell called CSLUsh (pronounced “slush”). CSLUsh incorporates the core technology modules with the well known, freely available, and easy-to-learn Tcl/Tk scripting language. Each C-language module is made available as a Tcl scripting command, and data are referenced as objects that can travel a network or be saved to disk in a device-transparent way. A CSLUsh application is built by using Tcl or Tk commands to call the core modules and manipulate the returned objects.

#### *2.2.3 Toolkit GUI Application Level*

At the highest level of the Toolkit are sophisticated GUI applications that allow rapid development of spoken language systems or the display and editing of speech data. These applications are written using CSLUsh and provide the user with an intuitive, powerful interface to the underlying speech technology. One such application is the Rapid Application Developer (RAD); RAD provides developers of speech applications with a graphical authoring environment for constructing interactive systems. With RAD, the developer can easily control Baldi's speech and appearance, modify the recognition vocabulary and parameters, employ clickable images, and play audio files.

A second application is SpeechView, which allows users to create new waveform and label files, display data (such as spectrograms) that are associated with a waveform, and modify existing waveforms and label files. Several independent waveform windows, each with zero or more spectrogram and label windows, may be displayed simultaneously within SpeechView for comparison and manipulation. Several spectrogram formats with user-defined signal processing and display options are available, and waveform sections corresponding to a phoneme or word label can be played back in isolation from adjacent phonemes or words.

Finally, it should be noted that individual researchers can modify any level of the Toolkit, adding or changing the C modules, CSLUsh scripts, or end-user applications to suit their own needs.

## 3. Recognition Framework

The method for training neural network recognizers using the Toolkit consists of executing a sequence of CSLUsh scripts using description files that specify aspects of the corpora, the training conditions, and the recognizer architecture. In order to train a new recognizer, the description files are created and the CSLUsh scripts are used to perform the steps of file collection, category mapping, data generation, data selection, network training, and network evaluation. The same scripts can be used to train task-specific or general-purpose recognizers, using one corpus or multiple corpora. Recognizers can be trained in different languages; to date, the authors are aware of the CSLU Toolkit being used to train recognizers in English, Italian, Korean, Mexican Spanish, and Vietnamese.

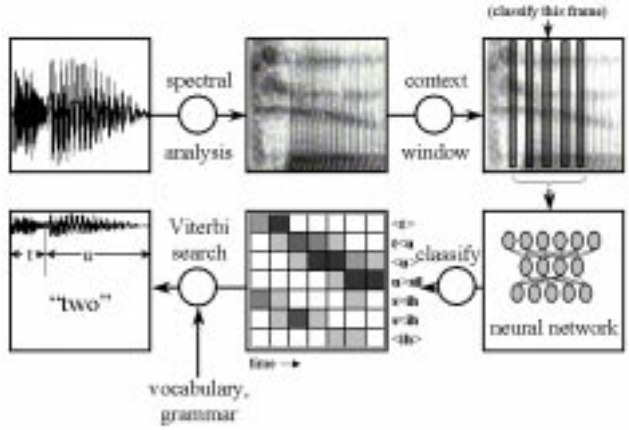


**Figure 1.** Graphical overview of the recognition process, illustrating recognition of the word “two”.

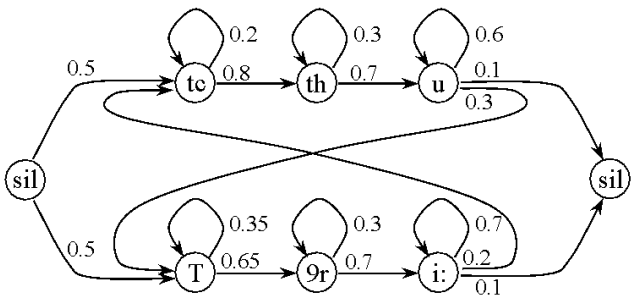


**Figure 2.** HMM state sequence for a two-word vocabulary.

The basic framework for the Toolkit's hybrid HMM/ANN speech recognition systems is illustrated in Figures 1 and 2. These systems use features that represent the spectral envelope (warped to emphasize the

perceptually-relevant aspects [8, 9]) and its energy given a fixed window size. These spectral features are computed at every 10-msec frame in the utterance and are input to the neural network for classification. The neural network receives not just the features for a given frame, but a set of features for the given frame and a fixed, small number of surrounding frames. This "context window" of features is used to provide the network with information about the dynamics of the speech signal.

At each frame, the neural network classifies the features in the context window into phonetic-based categories, estimating the probabilities of each category being represented by that set of features. The result of the neural network processing is a *C*x*F* matrix of probabilities, where *C* is the number of phonetic-based categories, and *F* is the number of frames in the utterance. The word or words that best match this matrix of probabilities is determined using a Viterbi search, given the vocabulary and grammar constraints. The search is usually thought of as traversing a state sequence (illustrated in Figure 2 with a simple two-word vocabulary), where each state represents a phonetic-based category, and there are certain probabilities of transitioning from one state to another.

The major difference between this framework and standard HMM systems is that the phonetic likelihoods are estimated using a neural network instead of a mixture of gaussians. Using a neural network to do this estimation has the advantage of not requiring assumptions about the distribution or independence of the input data, and neural networks easily perform discriminative training [10]. Also, neural networks can be used to perform recognition much faster than standard HMMs. A second difference is in the type context-dependent units; whereas standard HMMs train on the context of the preceding *and* following phonemes, our system splits each phoneme into states that are dependent on the left or right context, or are context independent.

## 4. Corpus

The OGI 30K Numbers corpus [11] was used for training, development, and testing the continuous digits recognizers. The data in this corpus were collected from thousands of people within the United States who recited their telephone number, street address, ZIP code, or other numeric information over the telephone in a natural speaking style. The data were collected from a large number of speakers from different backgrounds in different environments, and the corpus contains a noticeable amount of breath noise, glottalization, background noise (including music), and other "real-life" aspects that tend to make automatic speech recognition difficult. Of almost 15,000 utterances, approximately 6600 utterances have been transcribed and time-aligned at the phonetic level by professional labelers. The data have been labeled using Worldbet [12], and all phonetic symbols in this paper use the Worldbet notation. For the experiments reported here, we used those utterances that contain only the eleven digits (the numbers zero through nine, as well as "oh"). Before separating the data into training, development, and test sets, about 5% of the corpus was culled for independent testing and set aside. Three speaker-independent partitions were created from the remaining data: 3/5 for training (6087 files, of which 2547 were hand-labeled), 1/5 for development testing (2110 files), and 1/5 for final testing (2169 files). The development partition was further split into five sets of roughly 500 files each, and unless otherwise noted, the development results reported in this paper are for the first of these five sets (423 files[2]). This subset of the development set was used in order to be able to evaluate word-level performance with a large number of networks and search parameters in a reasonable amount of time.

## 5. Baseline System

The baseline system was trained using approximately the same method and parameters as the digits recognizer in the March 1998 release of the CSLU Toolkit.

One of the first steps in training the baseline system was to automatically map the hand-labeled phonetic symbols to a consistent set of symbols suitable for training. For example, the second vowel in the word "seven" was mapped to the single vowel /&/ (Worldbet notation for a neutral vowel), as there was a high degree of variability in the way that this vowel was labeled. In addition, the /oU 9r/ phonemes in the word "four" were merged into one />r/ phone, and the /kh s/ phonemes at the end of the word "six" were merged into one phone represented by the symbol /ks/. Finally, very short pauses (with duration less than 30 msec) were removed in order to improve the number of available contexts, and glottalization labels were merged into the surrounding vowels.

The system was trained to recognize context-dependent units. For left and right contexts, pauses and stop closures were mapped to the symbol /uc/ (unvoiced closure), and dentals (/th/, /s/, and the right half of /ks/) were mapped to the broad-category symbol /den/; otherwise the contexts were phoneme-specific. Each phoneme was split into one, two, or three sub-phonetic parts. The left part is dependent on the context of the preceding phoneme (or phonetic broad category), the center part (if any) is context independent, and the right part is dependent on the following phoneme (or phonetic broad category). Phonemes that remain as a one-part phoneme can either be context-independent (for example, /.pau/) or dependent on the following phoneme (for example, /th/).

The system was trained using 13 MFCC [9] features (12 cepstral coefficients and 1 energy parameter) plus their delta values, with a 10-msec frame rate. Cepstral-mean subtraction (CMS) [13] was performed, with the mean computed using all frames of data. The input to the network consisted of the features for the frame to be classified, as well as the features for frames at -60, -30, 30, and 60 msec relative to the frame to be classified (for a total of 130 input values). As many as 2000 samples per category were collected for training. Neural-network training was done with standard back-propagation on a fully-connected feed-forward network. The training was adjusted to use the negative penalty modification proposed by Wei and van Vuuren [14]. With this method, the non-uniform distribution of context-dependent classes that is

[2] In order to keep the development subsets speaker-independent, the subsets could not be split so that they each contained the same number of files; the first of these sub-sets happened to contain only 423 files.

dependent on the order of words in the training database is compensated for by flattening the class priors of infrequently occurring classes; this compensation allows better modeling for an utterance in which the order of the words can not be predicted.

Typical state duration models yield a Geometric distribution of state occupation, with rapidly decreasing likelihood of remaining in a given state as time increases. This shortcoming has been addressed before, by applying duration models based on Gamma distributions [15] as well as duration probabilities based directly on the training data [16]. In our implementation, transition probabilities were set to be all equally likely, so that no assumptions were made about the *a priori* likelihood of one category following another category. In order to make use of *a priori* information about phonetic durations, and to minimize the insertion of very short words, the search was constrained by specifying minimum duration values for each category, where the minimum value for a category was computed as the value at two standard deviations from the mean duration. During the search, hypothesized category durations less than the minimum value were penalized by a value proportional to the difference between the minimum duration and the proposed duration.

The grammar allowed any number of digits in any order, with optional silence between digits. In addition, a special word called "garbage" was allowed at the beginning and end of each utterance to account for out-of-vocabulary sounds. This "garbage" word consisted of a single context-independent category (also called "garbage"); the value for this category was not an output of the neural network, but was computed as the $N^{th}$-highest output from the neural network at each frame [17]. For example, if a neural network has three output values at one frame, {0.10, 0.60, 0.30}, and $N$ is 2, then "garbage" at that frame is 0.30 (the second-highest value). In this study, $N$ was set to 5; this value was empirically determined by varying $N$ until a roughly equal error rate between insertions and deletions was obtained on our task.

Training was done for 30 iterations, and the "best" network iteration was determined by word-level evaluation of the final 15 iterations on the development-set data. Then, each waveform in the same hand-labeled training set was then recognized using this best network, with the result constrained to be the correct utterance. This process, called "forced alignment," was used to generate time-aligned category labels. These force-aligned category labels were then used in a second cycle of training, and evaluation was repeated to determine the final digits network.

## 6. Experiments

We evaluated several aspects of training a digit recognition system, including the feature set, the amount of data used for training, the type of context-dependent categories, the values for duration limits, and the type of grammar. Once these experiments were completed, we trained a final system using forced alignment and the forward-backward method [18]. Each of these aspects is described in more detail below.

### 6.1. Features

As was noted by Barnard et al., "When speech recognition is performed with neural [networks], one should try to capture the important features of the desired output classes by features with invariant meaning. This will often require considerable knowledge of the speech problem…" [19]. Although our knowledge of human speech processing is limited, the first set of experiments was designed to determine which of the commonly used feature types, if any, are more suitable for classification by the neural network. We evaluated word-level performance of two common feature representations with and without their delta values.

Ten sets of features were evaluated: $13^{th}$-order MFCC with delta values (as used in the baseline system, referred to as *MFCC13D*), $13^{th}$-order MFCC with no delta values (*MFCC13*), $9^{th}$-order MFCC with and without delta values (*MFCC9D* and *MFCC9*), $13^{th}$-order and $9^{th}$-order PLP with and without delta values (*PLP13D*, *PLP13*, *PLP9D*, *PLP9*), a combination of $13^{th}$-order PLP and $13^{th}$-order MFCC (*PM13*), and a combination of $9^{th}$-order PLP and $9^{th}$-order MFCC (*PM9*).

The evaluation of the *combination* of PLP and MFCC features was motivated by the hypothesis that training with the two slightly different representations might provide somewhat more robustness.

The evaluation of each type of feature with and without delta values was motivated by the belief that the neural networks should, in theory, be able to learn the information provided by the delta values without having these values provided explicitly; the context window provided to the network already includes information about how the signal changes over time. By not using delta values, a smaller number of parameters needs to be estimated during training, which has the potential to make the training data easier to learn.

Two different cepstral orders (9 and 13) were used to test if the default value of 13 is an over-representation of the signal; with a sampling rate of 8000 Hz, there are on average only 4 formants, and the signal should be adequately represented by 2 cepstral coefficients per formant plus an additional coefficient to approximate the effect of the glottal source. Again, the smaller number of parameters required by a lower cepstral order may make the training data easier to learn.

All features were computed using RASTA [20] pre-processing, which filters out both very fast and very slow changes from the short-time spectrum. In an initial set of experiments [21], CMS was used with the MFCC parameters and RASTA was used with the PLP parameters to further distinguish the two methods. However, it was noted that the CMS processing was not pipelined, and so the entire waveform was used to compute the mean before doing subtraction. RASTA, on the other hand, does not look ahead in time when doing normalization. This may have given the MFCC/CMS combination a slight advantage over the PLP/RASTA combination. In a real-time system, where the utterance is processed in short segments, CMS will not have access to the entire waveform and may yield different results; RASTA, on the other hand, will give the same results. In order to have the results of these experiments generalize to real-time

systems, RASTA was used for all experiments reported in this paper.

### 6.2. Duration Limits

As noted in Section 5, duration limits are used to control the number of insertions of very short words. When there is a state transition during the Viterbi search, a check is made to see if the amount of time spent in that state is less than a minimum value. If the duration is less than the minimum, then a penalty is applied, with the penalty being linearly proportional to the difference between the proposed state duration and the minimum duration. (In a similar way, maximum duration limits are also applied, but their effect is not nearly as great as the minimum limits.) In the baseline system, the minimum and maximum values were set equal to two standard deviations from the mean duration. This was done to remove outlier durations that are not representative of their category (such as when phoneme deletion is not accounted for in a word model).

The formula for computing default minimum duration values was implemented based on educated guesses about the nature of phonetic durations. Our experiments on duration limits were motivated by the need for empirical justification for duration limit values. We evaluated each of the 10 recognizers trained with the features described above using four types of duration limits: with minimum duration values taken at two standard deviations from the mean (the default, referred to as *2SD*), from the 2nd percentile of all duration values (*2P*), from the 5th percentile of all duration values (*5P*), and from the 8th percentile of all duration values (*8P*).

The motivation for comparing the standard-deviation based limits with the percentile-based limits was related to assumptions about the distribution of the data. It was thought that although two standard deviations from the mean might be an appropriate value if the data are normally distributed, a percentile-based method may be a more reasonable method of removing outliers if the data are not normally distributed.

### 6.3. Grammar

The grammar for the baseline digits recognition system was defined as

**[*separator*] < *digit* [*silence*] > [*separator*]**

where square brackets (*[]*) indicate optional items, and angle brackets (<>) indicate one or more repetitions. The *separator* word was defined as

***silence* [*garbage*] *silence***

where *garbage*, as noted in Section 5, was a context-independent single-category word, with the category computed as the 5th-highest output from the neural network at each frame. This grammar allows optional silence between words and will be referred to as the *SIL* grammar.

We also investigated the use of a grammar that allows optional *garbage* to occur between words as well as silence. This grammar was defined as

**[*separator*] < *digit* [*separator*] > [*separator*]**

and will be referred to as the *GAR* grammar.

The motivation for evaluating both of these grammars was to test whether the optional pauses between words are modeled sufficiently well by the *silence* category, or whether a more complex model is needed. The risk of using the GAR grammar was that the number of deletions would increase, by having valid words recognized as *garbage*. On the other hand, it was thought that the GAR grammar might provide better modeling of the non-speech sounds that may occur between words.

### 6.4. Categories

As mentioned in Section 3, the networks are trained to recognize context-dependent categories. The contexts of each category can be either phoneme-specific or contain broad classes of similar phonemes; we evaluated all ten sets of features with both types of contexts: phoneme-specific (the default, *PHON*) and broad-class (*BC*). The broad classes of phonetic contexts are specified in Table 1. (The notation "_l" indicates that the context occurs on the left side of a phoneme; the "_r" notation indicates a context occurring on the right side of a phoneme.) The *PHON* recognizer had 218 outputs, and the *BC* recognizer had 149 outputs.

| name of broad class | phonemes in broad class |
|---|---|
| bck_l | oU w u |
| bck_r | oU w |
| den_l | s z th T ks |
| den_r | s z th T |
| lab | f v |
| ret_l | 9r >r |
| ret_r | 9r |
| sil | .pau tc kc .garbage |

**Table 1.** Broad class names and phonemes in each broad class. The "_l" and "_r" notation in the broad class name indicate whether the context occurs to the left or right of the phoneme.

The motivation for using the *PHON* set of categories was that the phoneme-specific differences in a particular context may provide additional information about the word. The motivation for using the *BC* set of categories was the belief that the phoneme-specific differences within one broad class are minimal, and that trying to determine minor phonetic differences in multi-speaker data might be difficult; the reduction in the number of categories to be learned may make training easier.

### 6.5. Amount of Data

We trained all of the systems described above on hand-labeled data using as many as 2000 samples per category. Using the best duration, garbage, and category models, we then trained networks on the 10 feature sets with all available hand-labeled data.

The motivation for this comparison was to estimate the effect on recognition performance when the amount of training data is increased.

### 6.6. Forced Alignment Training

Training on force-aligned data often results in improved recognizer performance because the sub-phonetic category alignments are determined from the speech data rather than by simply dividing the duration of a phonetic label into halves or thirds. Furthermore, it allows training on all

available training data, instead of just the available hand-labeled training data. The best network from Section 6 was used to force-align all training data, and a forced-alignment system, called *FA*, was trained from these data.

### 6.7. Forward-Backward Training

In addition to forced-alignment training, it is possible to use the forward-backward method to train neural networks [18]. We trained with this method for 45 iterations, using the FA recognizer as a starting point, and created a recognizer called *FB*. This was the recognizer on which test-set evaluation was done.

### 6.8. Evaluation Methodology

Due to the large number of possible combinations of tests, we conducted the evaluation using the following methodology:

1. Create the baseline system as described in Section 5, to confirm that the results of this recognizer are comparable to the results of the March 1998 CSLU Toolkit digits recognizer.
2. Train and evaluate the 10 sets of features with 2000 samples per category and the phoneme-specific category model, using the four duration models and two grammar models in combination (80 systems trained with up to 2000 samples per category, evaluated on the development set).
3. Using the best duration and grammar models from Step 2, train and evaluate the broad-class category model with the 10 sets of features (an additional 10 systems, trained with up to 2000 samples per category and evaluated on the development set).
4. Using the best category, duration, and grammar models from Step 3, train networks with the 10 feature sets on all available hand-labeled data, to study the effect of the amount of data (an additional 10 systems, trained with all hand-labeled training data and evaluated on the development set).
5. Using the best network from Step 4, do forced-alignment and forward-backward training (an additional 2 systems, trained with all training data and evaluated on the development set).
6. Select the best network from Step 5 and perform test-set evaluation (final evaluation of best system).

For evaluating the final recognition system and the baseline system on the test set, we computed the significance level using McNemar's test [22] (at the 5% level) and confidence intervals for both systems (at 95%). For computing the confidence intervals, we divided the test set into ten subsets (with approximately 217 utterances per subset) and determined the recognition accuracy on each of these subsets.

## 7. Results

The baseline system that we trained had word-level accuracy of 94.54% and sentence-level accuracy of 80.61%, which is comparable to the performance of the digits recognizer in the March 1998 release of the CSLU Toolkit, with 94.63% word accuracy and 82.27% sentence accuracy. A significant difference between the two systems can not be claimed for the sentence-level results at the 5% level ($P$=0.44).

For the next set of experiments, statistical significance testing was not done, because our goal was to measure improvement of a final system that employs all of the best available features; our goal was not to test whether the features that we selected as best can be considered significantly better[3]. Significance testing was done on the final system.

Figures 1 shows the results of the 10 feature sets evaluated on the four duration models with the SIL grammar. Figure 2 shows the results of the 10 feature sets evaluated on the four duration models with the GAR grammar. It is immediately obvious that the 2SD duration model yields lower word-level accuracy than the other duration models for all ten feature sets and two grammars. It can also be seen by comparing Figures 1 and 2 that the GAR grammar has better results than the SIL grammar for all ten features and four duration models. The average word-level results for the 2SD, 2P, 5P, and 8P duration models with the GAR grammar are 94.54%, 96.51%, 96.37%, and 95.88%, and so the GAR grammar with the 2P duration model was selected as the best combination.

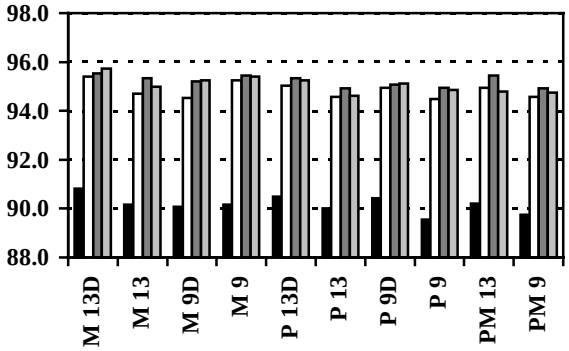


**Figure 3:** Word-level accuracy results for each of the 10 features using the four duration models with the SIL grammar. The horizontal axis codes are explained in Section 6.1. The black bar is for the 2SD limits, the white bar is for the 2P limits, the dark-gray bar is for the 5P limits, and the light-gray bar is for the 8P limits.

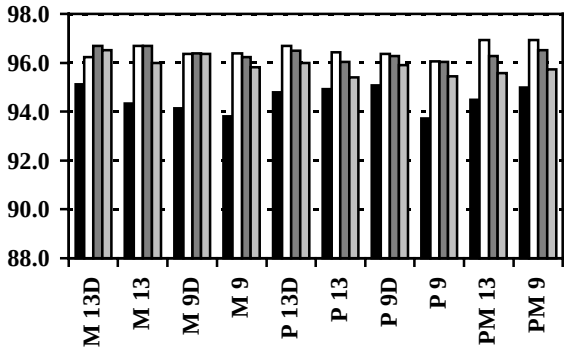


**Figure 4:** Word-level accuracy results for the same 10 feature sets as in Figure 1, using the same four duration models but with the GAR grammar.

[3] If statistical significance is not found, then significance can not be detected, which is different than the difference being insignificant.

Figure 5 shows the word-level performance of the BC and PHON category models for the ten feature sets (using the GAR grammar and 2P duration limits). The BC recognizer had 149 outputs, and the PHON recognizer had 218 outputs. It can be seen that the recognizers trained with the PHON categories had results consistently better than the recognizers trained using the BC categories, with an average 14% reduction in error. As a result, the PHON categories were selected as best.

Figure 6 compares the word-level performance of the recognizers trained using up to 2000 samples per category (177560 vectors, with an average of 814 samples per category) with the recognizers trained using all available hand-labeled data (402493 vectors, with an average of 1846 samples per category). The recognizers trained using all available data had, on average, a 7% reduction in error with 2.27 times the amount of training data. The feature set with the best performance on all training data was PLP13D, and so this set was used for subsequent experiments.

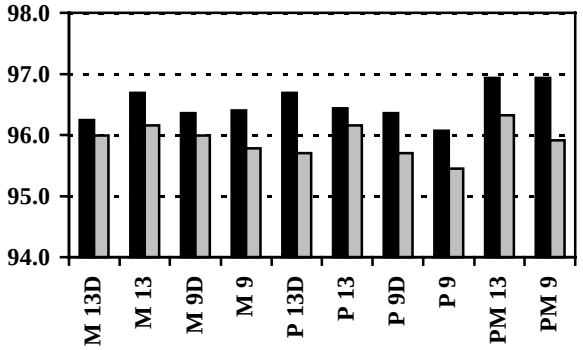


**Figure 5:** Word-level accuracy results for the same 10 feature sets as in Figure 1 (using the GAR grammar and 2P limits), comparing the PHON categories (dark bar) with the BC categories (light bar).

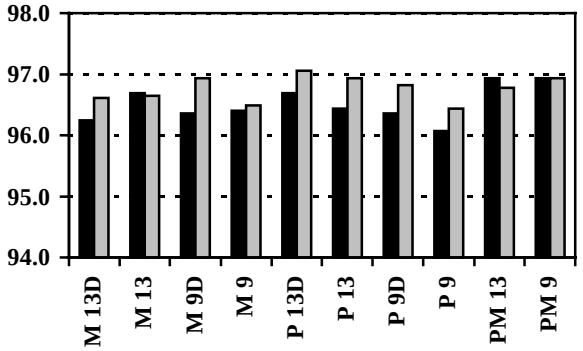


**Figure 6:** Word-level accuracy results for the same 10 feature sets as in Figure 1 (using the GAR grammar, 2P limits, and PHON categories), comparing the recognizers trained with 2000 samples per category (dark bar) and recognizers trained with all hand-labeled data (light bar).

Given the results of these experiments, the best set of parameters was determined to be the GAR grammar that allows optional garbage between words, the 2P duration limits which are computed from the 2nd percentile of duration values, the use of all available data, the PHON set of phoneme-specific categories, and 13th order PLP coefficients with their delta values. The system trained with these features on all available hand-labeled data had 97.06% word accuracy and 88.65% sentence accuracy on the development set.

For forced-alignment training, an average of 7676 samples per category were available. The development-set results from forced-alignment training were 97.56% (word) and 89.83% (sentence). Finally, the development-set results from forward-backward training were 97.56% (word) and 89.60% (sentence).

The results of test-set evaluation are summarized in Table 2. The 91.24% sentence-level result on 2169 files (12437 words) is significantly better than the 80.08% baseline result ($P<0.01$), and the confidence interval is ±0.73% for the baseline recognizer and ±0.71% for the new recognizer.

| System | Word Accuracy | Sentence Accuracy | Confidence Interval | Reduction in Error |
|---|---|---|---|---|
| Baseline | 94.65% | 80.08% | 94.65±0.73% | n/a |
| New | 97.52% | 90.36% | 97.67±0.71% | 56% |

**Table 2:** Test-set results for the baseline system and the new system, where the new system was trained with the set of best parameters as determined from the experiments in this paper. Evaluation was done on 2169 utterances (12437 words).

## 8. DISCUSSION

The results indicate that changing the grammar and duration limits had the greatest effect on recognizer performance, and that forced alignment of all data had the second-greatest effect.

The use of all available hand-labeled data, the type of categories, and the choice of features yielded smaller improvements. For the choice of features, the use of 13 cepstral coefficients and delta features often yielded a small improvement over the use of 9 coefficients and no delta features. This suggests that the extra cepstral coefficients and delta features do in fact provide useful information, although the differences in performance are overshadowed by other factors. The combination of PLP and MFCC features did not yield a noticeable, consistent improvement over the use of delta features with either MFCC or PLP.

The combination of all best feature sets and models, as well as forced-alignment training, did result in a statistically significant 50% reduction in test-set error. Rather than one factor being principally responsible for the improvement, it seems that the duration limits, grammar, amount of data, and forced-alignment training were all effective in contributing to the final performance. The forward-backward training did not improve performance in this experiment, but it should be noted that in a previous experiment on the same development set [21], forward-backward training did result in a 23% reduction in error.

The run-time complexity of the final system is the same as for the baseline system; both run in approximately real-time and have the same network configuration. Training time has been increased, simply because more training data is used for forced alignment and the forward-

backward method requires another cycle of network training; in many cases, the improvement in results would justify this additional training time.

For those who would like to replicate our results or try further experiments, both the OGI Numbers corpus and the CSLU Toolkit can be downloaded from http://cslu.cse.ogi.edu/ (free for academic use).

## 9. ACKNOWLEDGEMENTS

The authors would like to thank Johan Schalkwyk, Jacques de Villiers, Ben Serridge, Chris Covert, and the CSLU member companies. This work was sponsored in part by the National Science Foundation (grant numbers GER-9354959 and IRI-9614217); the views expressed in this paper do not necessarily represent the views of the NSF.

## 10. REFERENCES

[1] Sutton, S., Cole, R.A., de Villiers, J., Schalkwyk, J., Vermeulen, P., Macon, M., Yan, Y., Kaiser, E., Rundle, B., Shobaki, K., Hosom, J.P., Kain, A., Wouters, J., Massaro, D., and Cohen, M., "Universal Speech Tools: The CSLU Toolkit," ICSLP-98, vol. 7, pp. 3221-3224, Sydney, Australia, November 1998.

[2] http://cslu.cse.ogi.edu/tutordemos

[3] Cosi, P., Hosom, J.P., Shalkwyk, J., Sutton, S., and Cole, R.A., "Connected Digit Recognition Experiments with the OGI Toolkit's Neural Network and HMM Based Recognizers," IVTTA-ETWR-98, pp. 135-140, September 1998.

[4] http://cslu.cse.ogi.edu/toolkit

[5] Kaiser, E.C., Johnston, M., and Heeman, P.A., "PROFER: Predictive, Robust Finite-State Parsing for Spoken Language," ICASSP-99, vol. 2, pp. 629-632, Phoenix, AZ, March 1999.

[6] Black, A. and Taylor, P., "Festival Speech Synthesis System: System Documentation (1.1.1)," Human Communication Research Centre Technical Report HCRC/TR-83, Edinburgh, 1997.

[7] Massaro, D. W., *Perceiving Talking Faces: From Speech Perception to a Behavioral Principle*. MIT Press: Cambridge, MA, 1998.

[8] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," Journal of the Acoustical Society of America, vol. 87, no. 4, pp. 1738-1752, April 1990.

[9] Davis, S. and Mermelstein, P., "Comparison of Parametric Representations for Monosyllabic Word Recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-28, pp. 357-366, 1980.

[10] Bourlard, H., "Towards Increasing Speech Recognition Error Rates," Eurospeech'95, vol. 2, pp. 883-894, Madrid, Spain, September 1995.

[11] Cole, R.A., Fanty, M., Noel, M., and Lander, T., "Telephone Speech Corpus Development at CSLU," ICSLP-94, vol. 4, pp. 1815-1818, September 1994.

[12] Hieronymus, J, *ASCII phonetic symbols for the world's languages: Worldbet*. AT&T Bell Laboratories, Technical Memo, 1994.

[13] Furui, S., "Cepstral Analysis Techniques for Automatic Speaker Verification." *IEEE Transactions on Acoustic Speech and Signal Processing*, vol. 29, no. 2, pp. 254-272, 1981.

[14] Wei, W. and Van Vuuren, S., "Improved Neural Network Training of Inter-Word Context Units for Connected Digit Recognition," ICASSP-98, vol. 1, pp. 497-500, May 1998.

[15] Burshtein, D., "Robust Parametric Modeling of Durations in Hidden Markov Models", ICASSP-95, Detroit, pp. 548-551, 1995.

[16] Ferguson, J. D., "Variable Duration Models for Speech", Proceedings of Symposium on the Application of Hidden Markov Models to Text and Speech, pp. 143-179, Princeton, 1980.

[17] Boite, J.M., Bourlard, H., D'hoore, B., and Haesen, M., "A New Approach Towards Keyword Spotting," EUROSPEECH '93, vol. 2, pp. 1273-1276, September 1993.

[18] Yan, Y., Fanty, M. and Cole, R., "Speech Recognition Using Neural Networks with Forward-Backward Probability Generated Targets," ICASSP-97, vol. 4, pp. 3241-3244, April 1997.

[19] Barnard, E., Cole, R.A., Vea, M., and Alleva, F., "Pitch Detection with a Neural-Net Classifier," IEEE Transactions on Signal Processing, vol. 39, no. 2, pp. 298-307, February 1991.

[20] Hermansky, H. and Morgan, N., "RASTA processing of speech," IEEE Transactions on Speech and Acoustics, vol 2, no. 4, pp. 587-589, October 1994.

[21] Hosom, J. P., Cosi, P., and Cole, R. A., "Evaluation and Integration of Neural-Network Training Techniques for Continuous Digit Recognition," ICSLP-98, vol. 3, pp. 731-734, Sydney, Australia, November 1998.

[22] Gillick, L. and Cox, S.J., "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," ICASSP-89, pp. 532-535, 1989.