Connected Digit Recognition by Means of Reservoir Computing

Azarakhsh Jalalvand Fabian Triefenbach David Verstraeten Jean-Pierre Martens

ELIS, Ghent University, Sint-Pietersnieuwstraat 41, B-9000, Ghent, Belgium

Abstract

Most automatic speech recognition systems employ Hidden Markov Models with Gaussian mixture emission distributions to model the acoustics. There have been several attempts however to challenge this approach, e.g. by introducing a neural network (NN) as an alternative acoustic model. Although the performance of these so-called hybrid systems is actually quite good, their training is often problematic and time consuming. By using a reservoir – this is a recurrent NN with only the output weights being trainable – we can overcome this disadvantage and yet obtain good accuracy. In this paper, we propose the first reservoir-based connected digit recognition system, and we demonstrate good performance on the Aurora-2 testbed. Since RC is a new technology, we anticipate that our present system is still sub-optimal, and further improvements are possible.

Index Terms: speech recognition, reservoir computing, digits

1. Introduction

Although Hidden Markov Models (HMM) have proven their strong ability to model the speech acoustics for automatic speech recognition (ASR), they have regularly been challenged by alternative methods. Many of them try to alleviate the state-dependency hypothesis underlying the HMM paradigm. One such a suggestion is to model the dynamics and contextual dependencies in speech by means of a Recurrent Neural Network (RNN) [1]. It has been shown that RNN-based systems can indeed attain a good performance, but error back-propagation through time is a rather complex, critical and time consuming training method. The recently proposed Reservoir Computing (RC) paradigm [2, 3] may however provide an elegant solution to the training problem.

The basic idea of RC is that complex classifications can be performed by means of a pool of fixed (untrained) nonlinear interacting neurons, called the reservoir, and a set of trained linear classifiers that operate in the reservoir state space. The reservoir state is defined as the collection of the reservoir neuron outputs. The reservoir is nothing but a RNN and offers the capacity to model the dynamics of speech. The linear classifiers can be compared to the hyperplanes in the high-dimensional hidden feature space of an SVM [4], but with this difference that the latter space is obtained after training. The RC concept has already successfully been applied to different types of problems, including robot control, sequence generation and analysis, and even isolated spoken digit recognition [5, 6]. Recently, we have been able to demonstrate good English phoneme recognition capabilities as well [7].

In the present paper we propose the first RC-based connected digit recognizer, and we demonstrate good performance on the Aurora-2 testbed containing clean and noisy speech. In Section 2 we review the basics of RC, in Section 3 we describe the development of our RC-based recognizer of connected digits and in Section 4 we make an experimental assessment of the created system. The paper ends with some conclusions and ideas for future work.

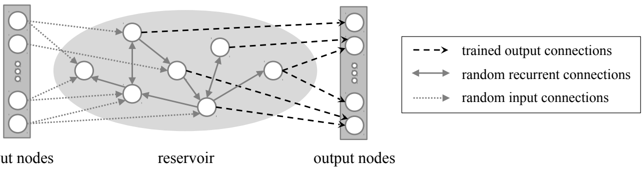

Figure 1: A basic RC system consists of a reservoir and a readout layer. The reservoir consists of nonlinear neurons with randomly fixed weights on the input and recurrent connections. Only the weights to the output nodes are being trained.

2. Reservoir Computing

A simple RC system (see Figure 1) consists of a reservoir and a set of read-out units. The reservoir is a pool of non-linear neurons which are connected to the inputs via input connections and to each other via recurrent connections. The weights on these connections are randomly generated and kept fixed throughout the system development. The so-called spectral radius (SR) defined as the largest eigenvalue (in absolute terms) of the recurrent weight matrix, controls the dynamics of the system. Each output node computes a linear function of the reservoir state, and the parameters of that function form the weights of the output connections. They are trained to achieve that a particular output node is high for observations of a particular class (e.g. phoneme or digit) and low for observations of any other class. Since the output node is linear, the output connection weights are obtained by linear regression. Since the output nodes ‘read’ the reservoir state, they are usually called the *read-outs*.

If K is the number of inputs, N the number of reservoir neurons (the reservoir size) and M the number of output neurons, the connection weights are collected in the matrices W_{in} ($K \times N$), W_{res} ($N \times N$) and W_{out} ($N \times M$). If u_t, x_t and y_t represent the values of the inputs, the reservoir outputs and the read-out nodes at time t, the RC system equations can be written as

$$x_{t+1} = f_{res}\left(W_{in}\, u_t + W_{res}\, x_t\right) \quad (1)$$

$$y_{t+1} = f_{out}(W_{out}\, x_{t+1}) \quad (2)$$

The functions f_{res} and f_{out} are the so-called activation functions of the reservoir and the output nodes. In this paper $f_{res}(x) = \tanh(x)$ and $f_{out}(x) = x$.

The output weights W_{out} are determined by means of ridge regression with the MSE as the objective criterion:

$$W_{out} = \arg\min_{W} \left(\frac{1}{N_{tr}} ||X\, W - D||^2 + \epsilon\, ||W||^2 \right) (3)$$

$$W_{out} = (X^T X + \epsilon\, I)^{-1}\, (X^T D) \quad (4)$$

The computed and desired output vectors are XW and D, while I is the unity matrix and N_{tr} the number of training examples. The term in ϵ is the regularization term.

To extend the integration of information over time, one can substitute the memoryless reservoir neurons by Leaky Integrator Neurons (LIN) [3]. Equation (1) then changes to

$$x_{t+1} = (1-\lambda)\, x_t + \lambda\, f_{res}\, (W_{in}\, u_t + W_{res}\, x_t) \tag{5}$$

with $0 \leq \lambda \leq 1$. The parameter λ (leak rate) encodes an integration time constant τ (in frames) via $\lambda = 1 - e^{-1/\tau}$.

3. Proposed method

In this section we describe how we constructed an RC-based recognizer of isolated and connected digits.

Originally, we adopted the approach of [5] in which there is a single read-out for each digit and for silence. However, like in HMM systems and like in [6], we generalized this approach to the case where each digit is modeled as a sequence of subword states, and each state is characterized by a read-out (see Figure 2). During operation, the likelihood of being in the state is determined by the read-out associated with that state.

The system proposed here can be viewed as a hybrid HMM system. In the first stage, a dynamic system (the reservoir) converts the input feature vector sequence into a sequence of vectors in a high-dimensional inner space (the reservoir state space). In the second stage, emission probabilities at a certain time (frame) are computed as a linear combination of the inner space variables at that time. The fundamental difference between the proposed system and a traditional hybrid system is that the mapping of the input features onto the inner space is traditionally trained whereas here it is completely random. We hope that the theoretical basis of SVMs – namely that an arbitrary binary classification in a well chosen (trained) high-dimensional inner space can be performed nearly optimally by means of a hyperplane – will also apply to the randomly created inner space.

In the subsequent sections we describe (1) the input feature sets we have used, (2) the reservoir weight generation scheme we have adopted, (3) the stochastic framework we have conceived for decoding the speech, and (4) the training procedure we have conceived for learning the read-out weights from non-segmented isolated and connected digit utterances.

3.1. Input features

As in most state-of-the-art recognizers, we also worked with the standard Mel Frequency Cepstral Coefficient (MFCC) setup, delivering 13 static (c_1,.., c_{12} and log$E$), 13 velocity and 13 acceleration features. In theory, the reservoir should be capable of modeling the short-term dynamics of the speech, and therefore would not need the non-static features. This is indeed confirmed experimentally to a large extent, but nevertheless we stick to the traditional 39 inputs because of two reasons: (1) since the input weights of the reservoir are fixed, more inputs do not raise the number of trainable parameters, and (2) adding the dynamic features does consistently offer a small benefit.

Since we also wanted to investigate the noise-robustness of our RC-based recognizer, we conducted our final experiments with the noise-robust MSVA features, proposed in [8] (MSVA stands for MFCC, Spectral Subtraction, Spectral Flooring and moving-average smoothing).

3.2. Reservoir weight generation

The recurrent weights of the reservoir are randomly drawn from a zero-mean Gaussian distribution with variance V. That variance is a control parameter that can be used to change the spectral radius (SR) of the reservoir. The SR is defined as the largest absolute eigenvalue of the recurrent weight matrix and is proportional to V. The SR is known to determine the dynamical excitability of the reservoir [2, 3].

Traditionally, the input weights are randomly drawn from a uniform distribution between $-$ISF and $+$ISF. The so-called input scaling factor (ISF) controls the relative importance of the inputs in the activation of the reservoir neurons. In [7] we refined this strategy by dividing the feature set in six sub-groups according to the dimensions (static, velocity, acceleration) and (MFCC, log-energy), and by using a separate input scaling factor for each sub-group. Here we adhere to this strategy. Note that traditional Gaussian mixture modeling does not require any input scaling at all since it is encoded in the variances of the individual mixtures. However, the sensitivity of a RC system to the choice of the input scaling seems to become marginal once the reservoir is big enough, as will be the case in our system.

3.3. A probabilistic framework

The construction of a probabilistic framework first of all involves the creation of a finite state automaton which represents what is spoken (during training) or what can be spoken (during recognition). It also takes into account the topologies of the acoustic models one wants to use for the digits and the silence. Figure 2 for instance shows the automaton that is used during recognition. The aim is to find the joint probability of observing the input sequence U along the state sequence Q through the automaton. The requested probability is computed as

$$P(Q,U) = \prod_{t=1}^{T} P(q_t|q_{t-1})\; P(u_t|q_t), \tag{6}$$

and $P(u_t|q_t)$ must be derived from the read-out vector y_t.

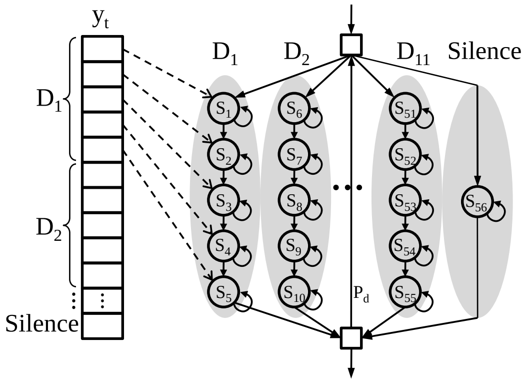

Figure 2: During recognition, the utterance model is a parallel loop of 11 digits and a silence, and each digit is modeled by a sequence of 5 states. On the left, the read-out layer is depicted and the arrows indicate the mapping of read-outs to states.

Suppose that the regression minimizes the mean squared error between the read-out vector y_t and the desired output vector d_t, and that all elements of d_t are -1, except the one corresponding to the desired state which is equal to +1. In that case, one can follow the derivations in [9] to show that under favorable conditions the rescaled read-out node $y'_{t,q} = 0.5 + 0.5 y_{t,q}$ will approximate the posterior probability vector $P(q|u_t)$, with q being any state of the automaton. In order to ensure that the probabilities are positive, we actually introduce the rescaled read-outs as

$$y'_{t,q} = \max(\frac{y_{t,q}+1}{2}, \delta) \qquad 0 < \delta \ll 1 \qquad (7)$$

and we compute the requested likelihood as

$$P(u_t|q_t) = \frac{P(q_t|u_t)}{P(q_t)}\, P(u_t) = \frac{y'_{t,q_t}}{P(q_t)}\, P(u_t) \qquad (8)$$

The prior probability $P(q_t)$ is obtained as the mean of $P(q_t|u_t)$ over all training frames.

Obviously, the probabilities $P(u_t)$ in (8) can be ignored during the probability maximization process as they are not a function of Q.

3.4. Training the system

The training procedure is organized in two phases. First we train a relatively small reservoir on the basis of isolated digit utterances. Then we train a larger system on the basis of all utterances, connected as well as isolated digits.

In the first phase, only isolated digit utterances are used because for these utterances it is possible to generate target labels of sufficient quality to determine the readout weights of a first reservoir. In order to obtain these targets, we first perform an energy-based segmentation of each utterance into silence-digit-silence and then presume a linear state progression inside the digit. In successive Viterbi iterations, new target labels are derived from the most likely state sequences that were produced using the actual reservoir, and the read-out weights are retrained on the basis of these labels. The process is continued for a couple of times (the number of iterations is not very critical since this is only the first phase of the training).

In the second phase, we also include connected digit utterances. These utterances are modeled as sequences of digits interleaved with optional silences, and surrounded by obligatory silences. Since we have much more utterances now, we can train a much larger reservoir. To start the training of that reservoir, we use the small reservoir emerging from phase 1 to derive initial target labels from a Viterbi alignment of the utterances with their models. From then on, the read-out weights are refined in a number of successive Viterbi iterations, each time using the latest reservoir as the acoustical model. The training is continued until saturation of the word error rate (WER) measured on a validation set is observed.

3.5. Recognition

During recognition, the model of Figure 2 is used as the utterance model. In order to control the trade-off between digit deletion and insertion errors, a word penalty P_d is assigned to the transition from the end state (bottom) to the start (top) state. The value of that penalty will be determined by means of recognition experiments on a validation set.

4. Experimental evaluation

All experiments are conducted on the Aurora-2 database [10]. This database contains clean and noisy utterances, sampled at 8 kHz and filtered with a G712/MIRS characteristic. There are 8440 clean training samples, 2412 of which contain only one digit. We have tested our systems on the clean test data (4004 utterances, 13159 digits) as well as on the noisy test sets A-C. The latter sets were created by artificially adding noise to the clean test data at Signal-to-Noise Ratios (SNR) between 20 and -5dB (see [10]). The vocabulary consists of the digits 0 to 9 and the letter 'o' (a substitute for 'zero').

During system development, only the clean data are used, and the input features are the 39 MFCCs. The training set is divided in a learning set (about 2/3 of the train data) and a validation set (the remaining 1/3 of the train data). The split was made such that there is no speaker overlap between the two sets. The topological parameters (e.g. the reservoir size) and the control parameters (e.g. how to scale the inputs) are optimized by performing experiments with different parameter values, and by selecting the parameter value which minimizes the word error rate (WER) obtained on the validation set.

Once all the control parameters are set, the final system is trained by repeating the training on the full training set. Since we have no validation set anymore in this stage, we just apply the number of iterations that we usually needed during the development phase. That actually means 4 iterations during phase 1 of the training and 5 more during phase 2.

4.1. Setting the control parameters

Following our previous work on phoneme recognition [7] we worked with only 50 recurrent connections per reservoir node. As we experienced before that the regularization constant and the safety parameter are not that critical if the reservoir size is large, we did not try to optimize them. They were fixed to $\epsilon = 0.001$ and $\delta = 0.002$ respectively, values that also work well for phoneme recognition. For the input scaling we also applied the same factors that we used for phoneme recognition.

The only parameters that were optimized for the digit recognition task are the spectral radius and the leak rate of the reservoir neurons. We did this because these parameters determine the dynamic behavior of the reservoir, and because the time scales of the phonemes and the digits are different. The optimization was performed with an isolated digit recognition system with a reservoir of 1000 nodes and a digit model with 3 states. We found a rather broad area in the (SR,λ) plane where the results remain pretty stable, and we finally selected SR = 0.8 and $\lambda = 0.35$ for all future experiments.

4.2. Setting the topological parameters

We performed three experiments to investigate two topological parameters : the reservoir size (number of reservoir nodes) and the number of states per digit (the same for all digits). We stick to the same number of states per digit because the reference HMM systems we want our system to compare with, also adopt this strategy. It would thus be unfair to optimize the number of states for each individual digit in our system. Recall that our silence model is always a single-state model.

Table 1 summarizes the most important results. In all experiments, the inter-digit transition probability was altered until a good balance between deletions and insertions was attained. The first experiment shows that the WER decreases with the reservoir size, but it starts to saturate as soon as the number of trainable parameters is larger than 100K. The second experiment reveals that doubling the number of states per digit from 5 to 10 is less effective than doubling the size of the reservoir (compare the improvements in I and II). On the other hand, the third experiment shows that a two-state system performs slightly worse than a five-state system with a comparable num-

Table 1: Validation results for systems with different reservoir sizes and number of states per digit.

Exp	nodes	states per digit	trainable params	WER in %
I	500	5	27500	2.71
	1000	5	55000	1.72
	2000	5	110000	1.29
	4000	5	220000	1.21
II	500	5	27500	2.71
	500	10	55000	2.33
	2000	5	110000	1.29
	2000	10	220000	1.27
III	4000	2	92000	1.35
	4000	5	220000	1.21

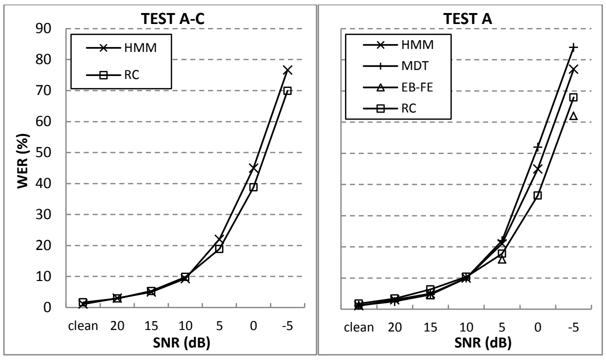

Figure 3: Recognition results (WER) as a function of SNR for the reference (HMM) and the reservoir system (RC). For test A, also results obtained with Missing Data Techniques (MDT) and Exemplar-Based Feature Enhancement (EB-FE) are shown.

ber of trainable parameters. The performance seems to saturate at five states per digit. The fact that we need less states than an HMM system (best results for 10 states) supports the claim that the reservoir does model dynamical properties of the speech.

5. Results on the noisy test sets

Once the optimal control parameters were fixed, including the inter-word penalty, we trained a new system with 4000 reservoir nodes and 5 states per digit. This time we used all the available clean training data and we worked with the MSVA input features. Figure 3 shows the average results (test sets A - C) of our system (RC) as a function of the SNR, as well as those of the reference HMM system described in [8]. The figures are taken directly from [8]. Apparently, the reservoir system competes well with the reference system, and for low SNRs it even yields a small benefit.

For test set A, more results are being published in the literature. That is why we also compared our system on this test set with two HMM systems recently described by Gemmeke [11] : one system in which a Missing Data Technique (MDT), namely imputation, is employed, and another in which Exemplar-based Feature Enhancement (EB-FE) is applied on the MFCC parameters. These figures confirm that our conceptually simple system achieves the same noise-robustness as these much more complex approaches.

6. Conclusion and future work

We have shown that Reservoir Computing, a fairly recent paradigm developed in machine learning, can be applied to create a good continuous digit recognizer. In combination with noise-robust features, the system competes favorably with a traditional HMM system, even if the latter is combined with complex noise suppression techniques such as Missing Data Imputation and Exemplar-based Feature Enhancement.

Since we have thus far only spent limited time on the development of our system, we may be able to further improve it soon. We could e.g. investigate stacked reservoir architectures like the ones we applied in our phoneme recognition work, or big reservoirs in combination with multiple read-out vectors. These read-out vectors would be defined in different subspaces of the reservoir state space, and read-outs from different vectors could be allowed to compete with each other.

7. Acknowledgments

The research leading to the results presented here has received funding from the European Community's Seventh Framework Program (FP7) under grant agreement 231267 "Self-organized recurrent neural learning of language processing" (ORGANIC).

8. References

[1] A. Robinson, "An application of recurrent neural nets to phone probability estimation," *IEEE Trans. on Neural Networks*, vol. 5, pp. 298–305, 1994.

[2] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the echo state network approach (48 pp)," German National Research Center for Information Technology, Tech. Rep., 2002. [Online]. Available: http://www.faculty.jacobs-university.de/hjaeger/pubs/ESNTutorialRev.pdf

[3] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, pp. 391–403, 2007.

[4] Q. Zhi-yi, L. Yu, Z. Li-hong, and S. Ming-xin, "A speech recognition system based on a hybrid hmm/svm architecture," in *Procs. International Conference on Innovative Computing, Information and Control*, 2006, pp. 100 –104.

[5] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Isolated word recognition using a liquid state machine," in *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN)*, 2005, pp. 435–440.

[6] M. Skowronski and J. Harris, "Automatic speech recognition using a predictive echo state network classifier," *Neural Networks*, vol. 20, no. 3, pp. 414–423, 2007.

[7] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 2307–2315.

[8] E. H. C. Choi, "A generalized framework for compensation of mel-filterbank outputs in feature extraction for robust asr," in *Interspeech*, 2005, pp. 933–936.

[9] M. Richard and R. Lippmann, "Neural net classifiers estimate posterior probabilities," *Neural Computation*, vol. 3(4), pp. 461–483, 1991.

[10] H. Hirsh and D. Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Procs. ISCA ASR*, 2000.

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