COMPARING NN PARADIGMS IN HYBRID NN/HMM SPEECH RECOGNITION USING TIED POSTERIORS

J. Stadermann, G. Rigoll

Institute for Human-Machine Communication
Munich University of Technology
Arcisstrasse 21, 80290 Munich, Germany
Phone:+49-89-289-{28319, 28541},
Email: {stadermann, rigoll}@mmk.ei.tum.de

ABSTRACT

Hybrid NN/HMM acoustic modeling is nowadays an established alternative approach in automatic speech recognition technology. A comparison of feed-forward and recurrent neural network topologies integrated in the tied-posteriors framework is presented. We give some insights in the training process of the networks estimating class posterior probabilities and show how the net's quality can be determined by introducing a new measurement prior to evaluating the complete ASR system. Finally we demonstrate the flexibility of the tied-posteriors framework by showing results for different context independent and context dependent acoustic models all based on the same system structure.

1. INTRODUCTION

A standard approach for designing acoustic models for speech recognition is the combination of hidden Markov models (HMMs) with (Gaussian) multi-mixture probability densities either using separate mixtures for each HMM state or using a set of shared (tied) mixtures. Improvements concerning speed and/or accuracy of the acoustic model can be achieved by using a combination of neural networks (NN) with hidden Markov models. There are several possibilities of integrating the NN in the speech recognizer architecture: The NN can replace the conventional feature extraction [1], it can be used as an advanced vector quantizer in discrete HMM systems [2] or it can be used as HMM output probability estimator [3, 4]. In the following we will only discuss the latter case. The first approach of probability estimation with a NN [3] uses a multi-layer perceptron (MLP) to estimate the posterior probability of each phoneme (given a feature vector) and use these probabilities directly in the HMM framework. This limits the HMM topology to one-state HMMs and allows context-dependent HMMs only with additional effort. The introduction of tied posterior probabilities [4] overcomes these limitations: The output of the NN is used as codebook of a semi-continuous HMM and the HMM output probabilities are composed of a weighted sum of the posterior probabilities entries. Starting from this approach presented - which uses an MLP for the probability computation - we introduce a *recurrent neural network (RNN)* as posterior probability estimator. From [5] it is known that a RNN-based recognition system produces error rates comparable to traditional HMM approaches. Using a RNN allows the implicit use of context information in the network as well as the reduction of the number of parameters compared to a multilayer perceptron (MLP). In contrast to [5] where more than one RNN is used in the complete system we take only a single RNN and can therefore achieve an even smaller system with nearly equal performance.
Furthermore we compare the performance of MLP and RNN tied-posterior systems if HMM state posterior probabilities are computed instead of phoneme posteriors. Finally we present the extension to triphone HMMs which is quite an easy task using tied-posteriors.

The remaining paper is organized as follows: Section 2 deals with the neural network paradigms that we have investigated, section 3 describes the architecture of the hybrid speech recognizer, section 4 shows the experiments' environment and methodology and section 5 presents the achieved results. Finally section 6 summarizes the achievements and gives some remarks on future work.

2. FEED-FORWARD AND FEEDBACK NEURAL NETWORKS

This section presents the architecture and the used algorithms of feed-forward and feedback neural networks (NN). Input to the networks is a feature vector consisting of 12 MFCC coefficients, the frame energy and the first and second derivative of these values (delta and acceleration features), resulting in a feature vector size of 39. In our frame-

work, one feature vector is computed every 10 ms from a data frame with a width of 20 ms. To improve the convergence speed, the features are normalized by subtracting the training set's mean value and dividing by the training set's variance [6].
Targets for the training process are obtained by a forced Viterbi alignment of the training data. In our experiments we have used two sets of target values:

- Viterbi alignment on phoneme level: Each NN output node represents one phoneme

- Viterbi alignment on HMM state level: Each NN output node represents one HMM state. For more details on the HMM topology, see section 3.

2.1. Feed-forward networks

The NN paradigm used here is a multi-layer perceptron (MLP) based on the one described in [3]. It is a three-layer network with full connection between nodes in adjacent layers. The input layer consists of the feature vector of frame t plus $2m$ additional vectors from neighboring frames resulting in an input vector $\vec{u} = (\vec{f}(t-m), \ldots, \vec{f}(t), \ldots, \vec{f}(t+m))^T$ The hidden layer uses the standard sigmoid nonlinearity. Since the output layer should represent probabilities we apply the softmax function to the output vector.
The MLP is trained with supervised back-propagation, the weights are updated after a block of training sentences is processed. Following [6, 7], we train the network to minimize the cross entropy between classes, to prevent overfitting we stop the training if an error measure[1] on a cross validation set is increasing. Optionally we can force the network to train a fixed number of iterations, before the cross validation test is performed.

2.2. Recurrent networks

Our recurrent neural network (RNN) is based on the one presented in [8], see figure 1. It is a partial recurrent network with separate output and feedback nodes, but with full connection between input, output and feedback nodes. In [9] is is stated that recurrent neural nets are as well as multi-layer perceptrons suitable for estimating class posterior probabilities. A RNN possesses the general advantage that context information of past time steps is implicitly stored in the feedback nodes. By delaying the network decision by m frames, we can incorporate information about future time frames, as well. So, applying feature vector $\vec{u}(t-m) = \vec{f}(t-m)$ to the input, we obtain the output $\vec{y}(t)$ and the feedback vector $\vec{x}(t+1)$. Similar to the MLP

[1]In our case, the frame error rate, described in section 4 is used.

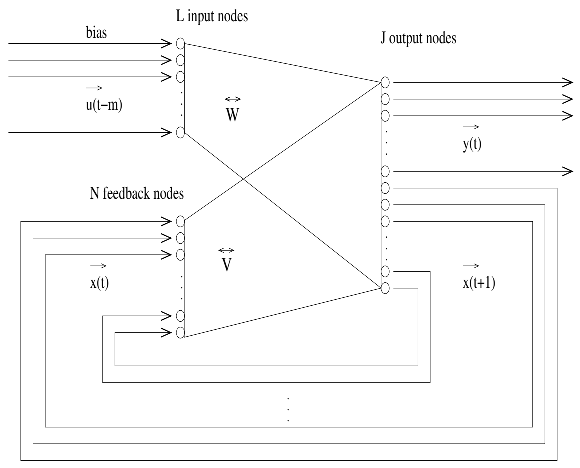

Fig. 1. Basic RNN architecture

we use a sigmoid function in the feedback nodes and a softmax function

$$y_i(t) = \frac{\exp(s_i(t))}{\sum_{\xi=1}^{J} \exp(s_\xi(t))} \tag{1}$$

in the output nodes.
Since we are dealing with a quite large training set (roughly 10^6 frames), the *back-propagation through time* (BPTT) algorithm was found to be the most suitable training algorithm for this task: In the first pass the network propagates through a block of training data and unfolds the network in time. The result is a virtual multi-layer perceptron with a number of hidden layers equal to the number of time steps and an output vector at each time step. A second pass is required to propagate the output errors back through time. The function to be optimized is again the cross entropy function and the stopping criterion is the same one as in the MLP case. Differing from the MLP training described above, we apply the RPROP [10] weight update strategy here. Parameters to be tuned are the number of feedback nodes, the block size used for BPTT and the number of frames after which a weight update takes place.

3. INTEGRATION OF THE NN IN THE HMM FRAMEWORK

As already stated in section 1, we are using the *tied-posteriors* approach presented in [4] to integrate the neural net in the HMM framework. The tied-posteriors approach is based on a (continuous) tied-mixture system where the output probabilities of each state are denoted as

$$p(\vec{x}|S_i) = \sum_{j=1}^{J} c_{ij} \cdot p(\vec{x}|j) \; , \tag{2}$$

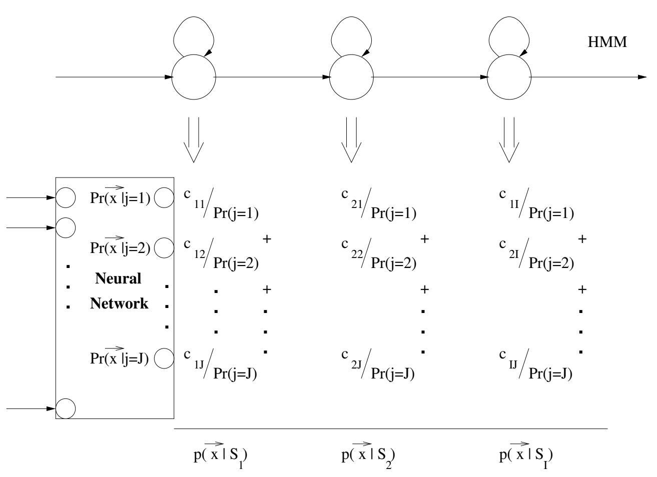

Fig. 2. HMM/NN acoustic model with $I = 3$ HMM states

where S_i is the HMM state and c_{ij} are the mixture coefficients. The idea is now to replace the probability density $p(\vec{x}|j)$ by the posterior probability $Pr(j|\vec{x})$ by using

$$p(\vec{x}|j) = \frac{Pr(j|\vec{x})p(\vec{x})}{Pr(j)} \tag{3}$$

Since $p(\vec{x})$ is independent of the HMM state S_i it can be omitted and (2) becomes

$$p(\vec{x}|S_i) \propto \sum_{j=1}^{J} c_{ij} \cdot \frac{Pr(j|\vec{x})}{Pr(j)} \tag{4}$$

The posterior probability $Pr(j|\vec{x})$ is estimated by the NN. As stated in section 2 the number of output nodes of the NN J is either equal to the number of phonemes or to the number of HMM states, the *a-priori* probability $Pr(j)$ can be estimated by counting the phoneme/state occurrences in the training set for both cases.

3.1. Monophone HMMs

All phoneme HMMs and the silence HMM are linear left-to-right models with 3 states. We use an additional HMM for short pauses between words that consists of one state. The overall number of monophone HMMs is 47 with 139 states in total. Figure 2 shows the principle set-up of the tied-posteriors system. The advantage of this approach is that the posterior probabilities are not directly connected to the HMM but only via the mixture coefficients c_{ij}. It is possible to change the number of network outputs (here from 47 to 139) without changing the HMM topology. On the other hand, the HMM topology can be changed - one state HMMs, 3-state HMMs, triphones - without changing the NN. The mixture coefficients c_{ij} are estimated using a standard maximum likelihood criterion and the Baum-Welch training algorithm.

3.2. Triphone HMMs

As mentioned before, introducing triphones produces no extra architectural effort in a tied-posteriors system. The number of HMMs increases from 47 to 10500, but according to eq. 4 the number of NN outputs stays at 47 and 139, respectively. We use word-internal triphones and start with a trained single-mixture monophone system as initialization. To reduce the number of parameters we use a modified tree-based clustering algorithm based on the one available in the Hidden-Markov model tool kit [11] applied to the weights c_{ij}. After tree-based clustering, the number of models is reduced to roughly 1000.

The language model used for the word recognition is the standard bigram language model that comes with the WSJ0 database.

4. EXPERIMENTAL ENVIRONMENT

All experiments have been evaluated on the WSJ0 speech data corpus with a closed vocabulary of 5000 words. The NN and the HMM have been trained on the speaker-independent training corpus which contains 7240 spoken sentences. The speech recognition performance of the complete system has been tested on the speaker-independent test from November 1992.
Additionally, we have computed the frame error rate (FER) of the neural network on a training set's sub set for cross validation during the NN training (724 sentences). The frame error rate is computed as follows:

- Take the index of the output node with the highest posterior probability

- Compare this index with the Viterbi alignment - if the index corresponds to the correct model count a correct frame otherwise count an error

From [12] it is known that a low FER does not automatically guarantee a low word error rate. The results in section 5 prove this fact, so we try to find another indicator to judge the NN's quality for speech recognition before training the HMMs. Here, we suggest to compute the NN's phoneme error rate (PER). The PER is computed by creating a sequence of phoneme symbols resulting from the sequence of highest NN posterior probabilities. Figure 3 illustrates the creation of the phoneme sequence. This phoneme sequence is compared to the training data with common dynamic programming techniques taken from [11]. We obtain an error

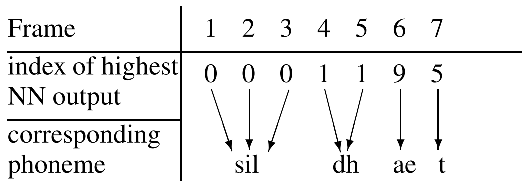

Fig. 3. Phoneme string created by the neural net

rate as follows:

$$\mathrm{PER} = 1 - \mathrm{Correctness}\,\text{(disregarding insertion errors)} \quad (5)$$

Confusions between the two output nodes representing *silence* and *short pause* are eliminated in the PER, since they do not influence the word error rate. Additionally, we count the absolute number of insertions, deletions and substitutions. These measures can tell us, if the NN tend to change its probability values quite rapidly between adjacent frames (high number of insertions) or if it shows some tendency to keep its current output state (low number of insertions). We found out, that the number of deletions and substitutions determines the NN's performance in the ASR system.
Finally, the word error rate (WER) ($\mathrm{WER} = 1 -$ Accuracy) of the complete hybrid system is evaluated on the given test set. As stated in section 3, the tied-posteriors approach allows arbitrary HMM topologies, so we present results for 3-state monophone models as well as 3-state triphone HMMs.

5. RESULTS

The first part of the results is a comparison of frame error rates (FER) obtained on our cross-validation set, followed by a table with phoneme error rates (PER) computed on the same set of data. The FER and PER of RNN139 and MLP139 (state alignment) are computed by adding the three state posteriors to get a phoneme posterior probability. This procedure allows to compare the nets at least to some extent with RNN47 and MLP47, respectively

The *weight factor* denotes the ratio between the number of weights compared to the first given MLP. The number of weights directly determines the speed of the NN, so a lower value is desirable. Since the RNN computes the weights at no extra cost[2], the weight factor is proportional to the gain in computing time. The NNs with 139 output nodes are trained using a HMM state alignment, the NNs with 47 output nodes use a phoneme alignment. The MLP uses 7 frames in the input layer (corresponding to $m = 3$, see section 2), the RNN takes one frame as input and delays its decision by 3 frames ($m = 3$).

[2]compared to a feed-forward net

neural net	**#output nodes**	**#input nodes**	**#weights (weight factor)**	**FER**
MLP47	47	273	321000 (1,0)	27,54%
RNN47	47	39	196680 (0,61)	25,64%
MLP139	139	273	413000 (1,29)	27,96%
RNN139	139	39	149260 (0,46)	27,71%

Table 1. Frame error rate (calculated from NN output)

neural net	**del.**	**subs.**	**ins.**	**err.**
MLP47	575	7466	50138	13,92%
RNN47	983	8401	32608	16,25%
MLP139	664	7620	47376	14,35%
RNN139	694	8530	45881	15,97%

Table 2. Phoneme error rate (calculated from NN output)

From table 1 the RNN paradigm is the clear winner regarding frame errors, but looking at table 3, the MLP still achieves better performance. Starting from the results in [12] we compute the PER as described in section 4. Here, a difference is observed: The RNN produces a much smaller number of insertions due to the lower FER. On the other hand, the error (disregarding insertions) is higher than in the MLP case. These two facts let us conclude that the MLP produces more errors during a phoneme segment, but assigns most segments to the correct posterior vector (with the correct phoneme getting the highest probability value). In contrast to that, the RNN produces little errors within a phoneme segment, but often assigns the wrong posterior vector to other segments. These segments get more or less lost in the HMM decoding step. A reason for this might be the RNN's built-in state-keeping behavior.

neural net	**del.**	**subs.**	**ins.**	**error**
MLP47	106	385	61	10,31%
RNN47	109	520	94	13,51%
MLP139	80	388	61	9,88%
RNN139	73	437	81	11,04%

Table 3. Word error rate *Monophones*

The RNN's higher word error rate in table 3 mainly stems from a higher number of (word) substitutions, a fact that is again a hint for our discussion of the PER results. The significantly better performance of RNN139 (state alignment) compared to RNN47 (phoneme alignment) shows us that increasing the RNN's number of feed-forward weights helps to overcome the explained problems.

neural net	**del.**	**subs.**	**ins.**	**error**
MLP47	100	343	42	9,15%
RNN47	122	461	64	12,09%
MLP139	82	334	42	8,56%
RNN139	80	418	70	10,52%

Table 4. Word error rate *Triphones*

Using tree-based clustered triphones (table 4), the gain in accuracy is evident throughout all net types. The rates might further improve if a better clustering is applied or if the recognition engine is tuned towards the triphone system (we use exactly the same set up as for the monophone case).

6. CONCLUSION

The tied-posteriors approach is presented emphasizing its flexibility to combine different neural net paradigms with varying HMM topologies. We compare feed-forward and recurrent neural networks estimating a varying number of posterior probabilities. A gain in terms of computation time is possible using recurrent networks due to the lower number of weights, but its inherent state-keeping attribute generated by the feedback nodes lowers the recognition accuracy. Increasing the number of forward weights improves the situation. Apart from the frame error rate we introduced a new measurement to rate the neural net's performance prior to training complete acoustic models. Our comparison using the WSJ0 task includes monophone and triphone acoustic models and both NN types achieve a performance comparable to standard Gaussian systems. Future work includes a closer look at the RNN's behavior to come up with the MLP results and the application of clustering techniques to adjust the number of output neurons in a systematic and optimal way.

7. REFERENCES

[1] Daniel Willett and Gerhard Rigoll, "Hybrid NN/HMM-Based Speech Recognition with a Discriminant Neural Feature Extraction," in *Advances in Neural Information Processing Systems (NIPS) 10*, Denver, Dec. 1997.

[2] Jörg Rottland, Christoph Neukirchen, and Daniel Willett, "Performance of Hybrid MMI-Connectionist/HMM Systems on the WSJ Database," in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Munich, Apr. 1997, pp. 1747–1750.

[3] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[4] Jörg Rottland and Gerhard Rigoll, "Tied Posteriors: An Approach for Effective Introduction of Context Dependency in Hybrid NN/HMM LVCSR," in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, June 2000.

[5] M. Hochberg, G. Cook, S. Renals, A. Robinson, and R. Schechtman, "ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System," in *Spoken Language Systems Technology Workshop*, 1995, pp. 170–176.

[6] Merten Joost and Wolfram Schiffmann, "Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUKFS)*, vol. 6, no. 2, pp. 117–126, 1998.

[7] P. Zhou and J. Austin, "Learning Criteria for Training Neural Network Classifiers," *Neural Computing and Applications*, vol. 7, pp. 334–342, 1998.

[8] A. J. Robinson, "An Application of Recurrent Nets to Phone Probability Estimation," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298–305, Mar. 1994.

[9] S. Santini and A. Del Bimbo, "Recurrent neural networks can be trained to be maximum A posteriori probability classifiers," *Neural Networks*, vol. 8, no. 1, pp. 25–29, 1995.

[10] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *IEEE International Conference on Neural Networks*, 1993.

[11] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, and S.J. Young, "The 1994 HTK Large Vocabulary Speech Recognition System," in *Proc. ICASSP*, Detroit, Michigan, 1995, pp. 73–76.

[12] Michael Shire, "Relating frame accuracy with word error in hybrid ann-hmm asr," in *Proc. EUROSPEECH*, Aalborg, Danmark, Sept. 2001.