A comparison of phoneme recognition for deep neural networks and recurrent neural networks

Axel Karlsson
axelkarl@kth.se

Yiru Lyu
yiru@kth.se

Abstract

Hidden markov models(HMM) have historically commonly been used for automatic speech recognition(ASR). More recently different deep learning models have been tried either by themselves for the same purpose or in combination with HMM models. In this paper the performance of different deep neural networks and recurrent neural networks are compared for ASR when they are combined with an HMM. The experiments show that the recurrent neural network generally performed better and that a model with one recurrent layer and two dense layers achieved a very good accuracy of 0.85 when classifying frame by frame at the phoneme level while still being much faster to train than more complicated recurrent models.

1 Introduction

As a traditional speech recognition method, the hybrid system composed of a deep neural network (DNN) and a Hidden markov models (HMM) has been playing a dominant role in the ASR field, and it still provides very good results [1]. Considering that speech data has rich temporal correlation, and recurrent neural network (RNN) has a strong advantage in dealing with sequential data, the RNN-based model has also begun to be applied to speech recognition [2]. Except for DNN and RNN, convolutional networks are also examined in such research [3].

More recently, end-to-end systems have also been reported on. In these cases, HMM is replaced by using networks with Connectionist Temporal Classification (CTC) object function [4, 5], so training an HMM can be avoided. The development of high performance computing and end-to-end deep learning approaches are providing better speech recognition results. Based on this, an attention-based mechanism have also been introduced [6], which makes good use of features from previous alignments.

The goal of this paper is to compare the performance of DNN and RNN layers in a hybrid system. In the following sections, different DNN and RNN implementations in combination with HMM for phoneme recognition will be compared and evaluated.

An example of previous research in this area is seen in [7]. That paper does describe the performance of RNNS and DNNs for ASR however it is focused on the evaluation of training principles and not evaluations of performance with regards to RNNs or DNNs. The results of that paper showed a marginally lower phone error rate for DNNs.

2 Background

2.1 Hidden Markov Model

In the experiments used for this paper viterbi approximation is used and defined as follows with the notations:

N: number of states
π_i: a priori probability at state s_i
$\alpha_i j$: transition probabilities from state s_i to state $s_i j$
$log\phi_j(x_i)$: observation log likelihood for each Gaussians

Forward probability formula:

$$log\alpha_0(j) = log\pi_j + log\phi_j(x_0)$$

$$log\alpha_n(j) = log(\sum_{i=0}^{M-1} exp(log\alpha_{n-1}(i) + log\alpha_{ij})) + log\phi_j(x_n)$$

Viterbi approximation formula:

$$logV_0(j) = log\pi_j + log\phi_j(x_0)$$

$$logV_n(j) = \max_{i=0}^{M-1}(logV_{n-1}(i) + log\alpha_{ij}) + log\phi_j(x_n)$$

2.2 Deep Neural Network

Deep neural networks consist of a multilayer perceptron with multiple layers of hidden units [8]. With notation such that the input layer is layer zero and the output layer is layer L in a $L+1$ layer DNN are the first layers:

$$v^l = f(z^l) = f(W^l v^{l-1} + b^l), for\, 0 < l < L$$

$$z^l = W^l v^{l-1} + b^l \in \mathbb{R}^{N_l \times 1}, W^l \in \mathbb{R}^{N_l \times N_{l-1}}, b^l \in \mathbb{R}^{N_l \times 1}$$

Different activation functions can be used such as the hyperbolic tangent function or the rectified linear unit function. For the multi-class classification tasks each output neuron represents a class $i \in \{1, ..., C\}$

2.3 Recurrent Neural Network

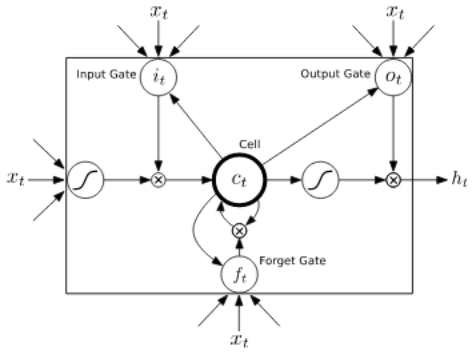
Recurrent neural network (RNN) is a kind of neural network that can be used to model time-dependent data. The specific manifestation is that the network will memorize the previous information and apply it to the calculation of the current output, ie, the nodes of the hidden layer not only connect to the input layer nodes but also output of hidden layer nodes at previous time step.

Long-short-term memory(LSTM) is a special type of RNN that can learn long-term dependency information. A common architecture is composed of a memory cell, an input gate, an output gate and a forget gate. [9, 10]

$x_t \in \mathbb{R}^d$: input vector
$f_t \in \mathbb{R}^h$: activation vector of the forget gate
$i_t \in \mathbb{R}^h$: activation vector of the input gate
$o_t \in \mathbb{R}^h$: activation vector of the output gate
$h_t \in \mathbb{R}^h$: output vector
$c_t \in \mathbb{R}^h$: cell state vector
$W \in \mathbb{R}^{h \times d}, U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^{h\}}$: The weight matrices and bias vector paramaters that are updated during the training of the models.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

Figure 1: LSTM cell [11]



3 Method

All data used in the models is from the TIDIGIGTS dataset[12].

- tidigits/disc_4.1.1/tidigits/train/ data is used for training
- tidigits/disc_4.2.1/tidigits/test/ data is used for testing.

Furthermore training is also split into a training and a validation set where six male and six female speakers are used for validation. The features are then transformed into their dynamic representation by stacking seven of them symmetrically for each utterance and timestep around the current timestep. Feature normalization is done by normalizing over the whole training set.

As a comparison/baseline, we use the combination of DNN and HMM to decode input features after feature extraction. In this process, we tried different DNN system parameters and chose suitable ones, such as learning rate and optimization method. In the following part of the experiment, we can see that for different input characteristics and network structure, we have established four DNN control groups. Our goal is to compare the performance differences between the RNN and DNN used for feature decoding in similar situations. Therefore, we replace the DNN in the speech recognition process with the RNN, that is, replace the Dense network layers with the LSTM network layers. Since the LSTM requires the input data to contain timestep, we perform a translation on the time axis of all the features on the basis of the original data, that is, each frame feature is associated with three frames before and after, so that each input feature obtained has a dimension of seven Timestep.

Different models for DNN and HMM were built where different hyperparameters were tested. The tested hyperparameters are shown below in the experiments section.

4 Experiments

As shown in the Table below, DNNs and RNNs with different settings are evaluated based on our prepared datasets. Variates including network structure, number of hidden layers, input feature type.

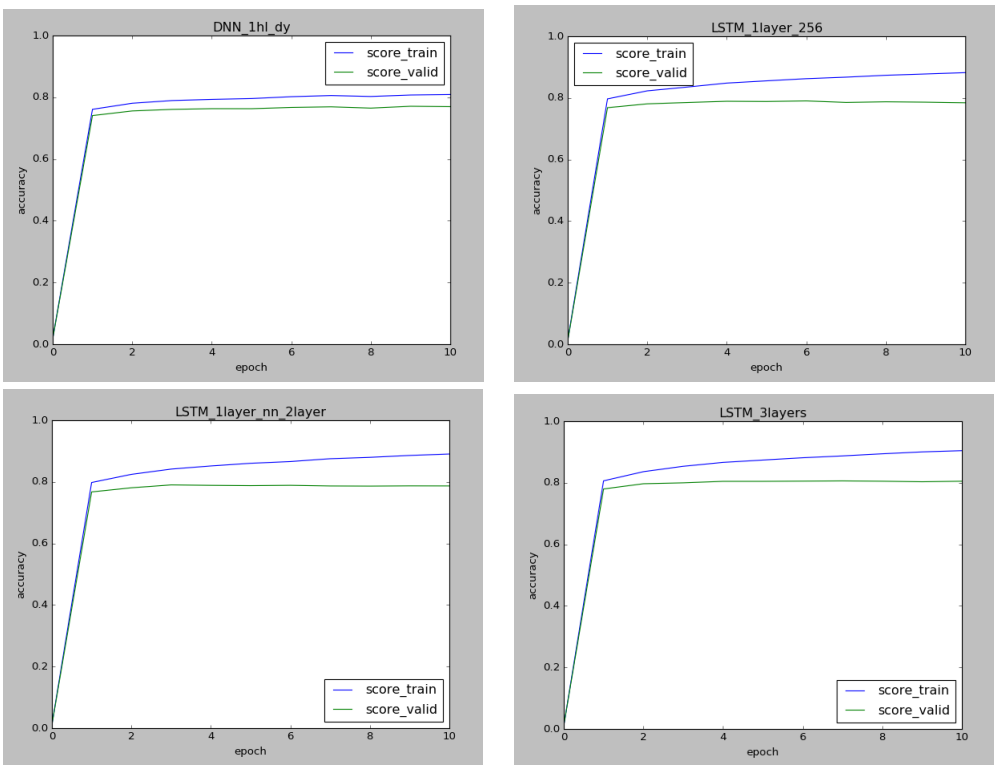
Network	Train loss	Train acc	Valid loss	Valid acc
DNN_1hl	1.3357	0.5965	1.3485	0.5922
DNN_1hl_dy	0.7775	0.7448	0.8115	0.7315
DNN_1hl_dy_mspec	0.7263	0.7649	0.7562	0.7527
DNN_3hl_dy	0.6894	0.7729	0.7495	0.7513
LSTM_1l	0.3587	0.8819	0.7433	0.7865
LSTM_1l_mspec	0.3787	0.8749	0.6946	0.7934
LSTM_1l_dy	0.2147	0.9249	0.6511	0.8184
LSTM_1l+DNN_2l	0.3763	0.8737	0.7088	0.7955
LSTM_3l	0.2872	0.9044	0.6974	0.8051

Dense layers with 256 units for each hidden layer are applied to all the DNNs. All networks are trained with stochastic gradient descent, with learning rate 0.02 and momentum 0.9. The activation function for hidden layers used here is ReLu and softmax for the output layer. Some tests were done where ReLu and sigmoid activation functions were compared for very simple networks however since ReLu gave better results in those simple cases all experiments discussed in this paper use ReLu as the activation function. As to the LSTM layers, the optimizer used is adam, which provides a relatively better training result. The number of nodes in hidden layers are 256 each, the same as DNNs, and the output layer of the network is also same as DNNs. All the results in the table are calculated after 10 epochs of training.

5 Results

From the result of "DNN_1hl_dy" and "LSTM_1l", we can find that the latter network performs better both in training and validation accuracy. In these two cases, the DNN network take the dynamic features as input, which include time evolution with features at time [n-3:n+3]. The LSTM network also include the same range of time information by using time steps of [n-3:n+3], but gives better training results. If we use the dynamic feature as the input of LSTM, we can get even better results, as "LSTM_1l_dy" shows.

Since we using DNN or LSTM we do not require the independence between feature dimensions, we also used mspec as input feature for both networks. Comparing "DNN_1hl_dy" with "DNN_1hl_dy_mspec" and "LSTM_1l" with "LSTM_1l_mspec", it is obvious that mspec performs better than lmfcc. This is understandable because lmfcc contains less feature information than mspec.



The graphs show training and validation accuracy at a certain epochs for the different networks Top-left: DNN with 1 dense hidden layer, using dynamic lmfcc features Top-right: RNN with 1 LSTM layer, using lmfcc features
Bottom-left: RNN with 1 LSTM layer and 2 dense hidden layers, using lmfcc features
Bottom-right: RNN with 3 LSTM layers, using lmfcc features
Note that the validation accuracy stagnates while the training accuracy continues to increase due to overfitting.

Detailed evaluations are done for the classification performance on the test set with our selected DNN and RNN networks.

DNN_3hl_dy, Frame by frame at the state level: The classification accuracy on the whole test dataset is 0.74107375570

DNN_3hl_dy, Frame by frame at the phoneme level: The classification accuracy on the whole test dataset is 0.839466435802

LSTM_1l_DNN_2l, Frame by frame at the state level: The classification accuracy on the whole test dataset is 0.770662875389

LSTM_1l_DNN_2l, Frame by frame at the phoneme level: The classification accuracy on the whole test dataset is 0.85487428406

6 Discussion and Conclusions

1. When implementing the RNN, we extend the time step of the dataset by shifting all the feature vectors along the frame axis, that is, using features at time [n-3:n+3] as the time steps of the input features. This helps increase accuracy in practice but may cause errors at the frames where pronunciation changes. As we have analysed, if we can extend the time steps more detaily by shifting the features for each phoneme, we may get more accurate result.

2. With the same number of layers and the same number of nodes in each layer, LSTM trains much slower than DNN. Besides, in our experiments, one LSTM layer with 2 dense hidden

layers provides a similar training and validation accuracy compared to 3 LSTM layers, but the former structure trains much faster.

3. Whether we use a DNN or LSTM to solve classification problems, we cannot avoid using HMM to generate the targets. However, this process is not reliable. The objective function used to train the networks is different from the whole transcription accuracy. In our experiments, we assumed a SGMM as the emission probability of HMM. More commonly, the output of the DNN or RNN will be formulated as the emission probability, which will lead to an iterative procedure. Based on these two main problems, end-to-end RNN has been used in speech recognition in [4]. Their system is a combination of the deep bidirectional LSTM and the Connectionist Temporal Classification object function. The network is trained directly on the text transcripts and no forced alignment is required to provide training targets.
4. The reason this topic of research(Comparison of DNN and RNN performance) was selected was due to interest in how the temporal behaviour of RNNs would affect performance for ASR. Before conducting the experiments it was hypothesised that this would positively affect performance since time was assumed to be an important dimension in ASR. The final results showed that this previous assumption was correct and that creating a model that takes the temporal aspects of ASR into consideration has positive effects.

References

[1] Edmondo Trentin and Marco Gori. A survey of hybrid ann/hmm models for automatic speech recognition. 37:91–126, 04 2001.

[2] A. Graves, N. Jaitly, and A. r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Dec 2013.

[3] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, Oct 2014.

[4] A Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. 5:1764–1772, 01 2014.

[5] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. 12 2015.

[6] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Y Bengio. Attention-based models for speech recognition. 06 2015.

[7] Oriol Vinyals, Suman V. Ravuri, and Daniel Povey. Revisiting recurrent neural networks for robust asr. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4085–4088, 2012.

[8] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer, London, 2015.

[9] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 1999.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

[11] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[12] R Gary Leonard and George Doddington. Tidigits speech corpus. *Texas Instruments, Inc*, 1993.

7 Appendix with changes from peer review

The peer review feedback that suggested changes are listed here followed by short explanations of how the suggested change was done or explanations of why it was not done.

- Does not really include any previous work:
An additional reference is added and discussed at the end of the introduction in order to make sure that previous work is considered in the paper.

- In abstract the results need to presented with values, like what your best accuracy was and what network:
This has been added to the abstract.

- On page 4 you describe the network settings with activation function being relu and so forth. Might be ease the read by moving this to section 3 method instead. Did you use any other activation functions?
We argue that the things described here are related to the particular experiments run for this paper and thus belong in the experiments session. So we did not make this suggested change. However the description of use of activation functions in the experiments was clarified due to the question.

- I think the discussion should be extended a bit, maybe discuss what results you expected, why and how it compares to what you got.
This is added at the end of the Discussion and Conclusions section

- In the discussion you maybe wanna take up the fact that the training accuracy increases but the validation accuracy seems to be stagnant.
This is added as a comment below the description of what the graphs depict in the results section.