Deep Bidirectional LSTM (DBLSTM) recurrent neural networks have recently been shown to give state-of-the-art performance on the TIMIT speech database. However, the results in that work relied on recurrent-neural-network-specific objective functions, which are difficult to integrate with existing large vocabulary speech recognition systems. This paper investigates the use of DBLSTM as an acoustic model in a standard neural network-HMM hybrid system. We find that a DBLSTM-HMM hybrid gives equally good results on TIMIT as the previous work. It also outperforms both GMM and deep network benchmarks on a subset of the Wall Street Journal corpus. However the improvement in word error rate over the deep network is modest, despite a great increase in frame-level accuracy. We conclude that the hybrid approach with DBLSTM appears to be well suited for tasks where acoustic modelling predominates. Further investigation needs to be conducted to understand how to better leverage the improvements in frame-level accuracy towards better word error rates.

Index Terms — DBLSTM, HMM-RNN hybrid

1. INTRODUCTION

Deep Bidirectional LSTM was recently introduced to speech recognition, giving the lowest recorded error rates on the TIMIT database. In that work the networks were trained with two end-to-end training methods designed for discriminative sequence transcription with recurrent neural networks, namely Connectionist Temporal Classification and Sequence Transduction. These methods have several advantages: they do not require forced alignments to pre-segment the acoustic data, they directly optimise the probability of the target sequence conditioned on the input sequence, and (especially in the case of Sequence Transduction) they are able to learn an implicit language model from the acoustic training data. However neither method can readily be integrated into existing large vocabulary speech recognition systems, which were designed around the GMM-HMM paradigm. In particular, it is not straightforward to combine them with word-level language models, which play a vital role in real-world tasks.
are given in Section 7.

2. NETWORK ARCHITECTURE

Given an input sequence \( x = (x_1, \ldots, x_T) \), a standard recurrent neural network (RNN) computes the hidden vector sequence \( h = (h_1, \ldots, h_T) \) and output vector sequence \( y = (y_1, \ldots, y_T) \) by iterating the following equations from \( t = 1 \) to \( T \):

\[
\begin{align*}
    h_t & = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\
    y_t & = W_{hy}h_t + b_y
\end{align*}
\]

where the \( W \) terms denote weight matrices (e.g. \( W_{xh} \) is the input-hidden weight matrix), the \( b \) terms denote bias vectors (e.g. \( b_h \) is hidden bias vector) and \( \mathcal{H} \) is the hidden layer function.

\( \mathcal{H} \) is usually an elementwise application of a sigmoid function. However we have found that the Long Short-Term Memory (LSTM) architecture [11], which uses purpose-built memory cells to store information, is better at finding and exploiting long range context. Fig. 1 illustrates a single LSTM memory cell. For the version of LSTM used in this paper [12] \( \mathcal{H} \) is implemented by the following composite function:

\[
\begin{align*}
    i_t & = \sigma (W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
    f_t & = \sigma (W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
    c_t & = f_t c_{t-1} + i_t \tanh (W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
    o_t & = \sigma (W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
    h_t & = o_t \tanh (c_t)
\end{align*}
\]

where \( \sigma \) is the logistic sigmoid function, and \( i, f, o \) and \( c \) are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the hidden vector \( h \). The weight matrices from the cell to gate vectors (e.g. \( W_{ci} \)) are diagonal, so element \( m \) in each gate vector only receives input from element \( m \) of the cell vector.

One shortcoming of conventional RNNs is that they are only able to make use of previous context. In speech recognition, where whole utterances are transcribed at once, there is no reason not to exploit future context as well. Bidirectional RNNs (BRNNs) [13] do this by processing the data in both directions with two separate hidden layers, which are then fed forwards to the same output layer. As illustrated in Fig. 2, a BRNN computes the forward hidden sequence \( \overrightarrow{h} \), the backward hidden sequence \( \overleftarrow{h} \) and the output sequence \( y \) by iterating the backward layer from \( t = T \) to 1, the forward layer from \( t = 1 \) to \( T \) and then updating the output layer:

\[
\begin{align*}
    \overrightarrow{h}_t & = \mathcal{H}(W_{xh}x_t + W_{h\overleftarrow{h}}h_{t-1} + b_{\overleftarrow{h}}) \\
    \overleftarrow{h}_t & = \mathcal{H}(W_{xh}x_t + W_{h\overrightarrow{h}}h_{t+1} + b_{\overrightarrow{h}}) \\
    y_t & = W_{hy}^{\overrightarrow{h}} h_t + W_{hy}^{\overleftarrow{h}} h_t + b_y
\end{align*}
\]

Combining BRNNs with LSTM gives bidirectional LSTM [14], which can access long-range context in both input directions.

A crucial element of the recent success of hybrid systems is the use of deep architectures, which are able to build up progressively higher level representations of acoustic data. Deep RNNs can be created by stacking multiple RNN hidden layers on top of each other, with the output sequence of one layer forming the input sequence for the next, as shown in Fig. 3. Assuming the same hidden layer function is used for all \( N \) layers in the stack, the hidden vector sequences \( h^n \) are iteratively computed from \( n = 1 \) to \( N \) and \( t = 1 \) to \( T \):

\[
\begin{align*}
    h^n_t & = \mathcal{H}(W_{h^{n-1}h}h^{n-1}_t + W_{h^n}h^n_{t+1} + b^n_h)
\end{align*}
\]

where we define \( h^0 = x \). The network outputs \( y_t \) are

\[
\begin{align*}
    y_t & = W_{h^ny}h^N_t + b_y
\end{align*}
\]

Deep bidirectional RNNs can be implemented by replacing each hidden sequence \( h^n \) with the forward and backward sequences \( \overrightarrow{h}^n \) and \( \overleftarrow{h}^n \), and ensuring that every hidden layer receives input from both the forward and backward layers at the level below. If LSTM is used for the hidden layers we get deep bidirectional LSTM, as illustrated in Fig. 4.
3. NETWORK TRAINING

Network training follows the standard approach used in hybrid systems [4]. Frame-level state targets are provided on the training set by a forced alignment given by a GMM-HMM system. The network is then trained to minimise the cross-entropy error of the targets using a softmax output layer with as many units as the total number of possible HMM states. At decoding time, the state probabilities yielded by the network are combined with a dictionary and language model to determine the most probable transcription. For a length $T$ acoustic sequence $x$ the network produces a length $T$ output sequence $y$, where each $y_t$ defines a probability distribution over the $K$ possible states: that is, $y_t^k$ (the $k^{th}$ element of $y_t$) is the network’s estimate for the probability of observing state $k$ at time $t$ given $x$. Given a length $T$ state target sequence $z$ the network is trained to minimise the negative log-probability of the target sequence given the input sequence:

$$- \log \Pr(z|x) = - \sum_{t=1}^{T} \log y_t^z_t$$  \hspace{1cm} (13)

Which leads to the following error derivatives at the output layer

$$- \frac{\partial \log \Pr(z|x)}{\partial \hat{y}_k^t} = y_t^k - \delta_{k,z_t}$$ \hspace{1cm} (14)

where $\hat{y}_t$ is the vector of output activations before they have been normalised with the softmax function. These derivatives are then fed back through the network using backpropagation through time to determine the weight gradient.

When training deep networks in hybrid systems with stochastic gradient descent it has been found advantageous to select minibatches of frames randomly from the whole training set, rather than using whole utterances as batches. This is impossible with RNN-HMM hybrids because the weight gradients are a function of the entire utterance.

Another difference is that hybrid deep networks are trained with an acoustic context window of frames to either side of the one being classified. This is not necessary for DBLSTM, since it is as able to store past and future context internally, and the data was therefore presented a single frame at a time.

For some of the experiments Gaussian noise was added to the network weights during training [15]. The noise was added once per training sequence, rather than at every timestep. Weight noise tends to 'simplify' neural networks, in the sense of reducing the amount of information required to transmit the parameters [16, 17], which improves generalisation.

4. TIMIT EXPERIMENTS

The first set of experiments were carried out on the TIMIT [18] speech corpus. Their purpose was to see how hybrid training for deep bidirectional LSTM compared with the end-to-end training methods described in [1]. To this end, we ensured that the data preparation, network architecture and training parameters were consistent with those in the previous work. To allow us to test for significance, we also carried out repeated runs of the previous experiments (which were only run once in the original paper). In addition, we ran hybrid experiments using a deep bidirectional RNN with tanh hidden units instead of LSTM.

The standard 462 speaker set with all SA records removed was used for training, and a separate development set of 50 speakers was used for early stopping. Results are reported for the 24-speaker core test set. The audio data was preprocessed using a Fourier-transform-based filterbank with 40 coefficients (plus energy) distributed on a mel-scale, together with their first and second temporal derivatives. Each input
vector was therefore size 123. The data were normalised so that every element of the input vectors had zero mean and unit variance over the training set. All 61 phoneme labels were used during training and decoding (so \( K = 61 \)), then mapped to 39 classes for scoring \([19]\). All experiments were repeated four times with different random initialisations, and results are quoted as the mean \( \pm \) the std. dev.

Table 1 shows the phoneme error rate (PER) for DBLSTM trained with the two methods described in \([1]\): Connectionist Temporal Classification (‘CTC’) and Sequence Transduction (‘Transducer’). Both networks consisted of five bidirectional hidden levels, each containing two LSTM layers of 250 cells, along with a size 62 softmax output layer (one unit for each phoneme, plus an extra blank unit). The sequence transduction network had an additional phoneme prediction network with a single hidden layer of 250 LSTM cells, and an output network with a single hidden layer of 250 \( \tanh \) units. The CTC network had approximately 6.8M weights and the Transducer network had approximately 7.4M. All networks were trained using stochastic gradient descent, with learning rate \( 10^{-4} \), momentum 0.9 and random initial weights drawn uniformly from \([−0.1, 0.1]\). The CTC networks were first trained to convergence with no noise, then retrained with weight noise (std. dev. 0.075). The Transducer networks were initialised with the weights of the CTC networks after retraining with noise. The Transducer phoneme error rate of 18.07 \( \pm \) 0.24 is consistent with the single result of 17.7 recorded in \([1]\). Indeed, the single best Transducer run in this paper (the one achieving lowest PER on the development set) also returned 17.7 on the test set.

For hybrid training on TIMIT a phonetic dictionary was used, with three states per phoneme, giving 183 target states in total. A biphone language model was estimated on the training set, and a simple GMM-HMM system was used to provide forced alignments. The posterior state probabilities provided by the networks were not divided by the state occupancy priors, as this has been found to make no difference on TIMIT \([6]\). Table 2 shows the phoneme error rates for hybrid training with DBLSTM and Deep Bidirectional RNN (DBRNN), along with the frame error rate (FER) and cross-entropy error (CE) in units of nats per frame. The DBLSTM networks had the same architecture as the CTC networks described above, except that the output layer had 183 units (one for each HMM state). As before, each randomly initialised LSTM network was first trained to convergence, then retrained with weight noise. The DBRNN network had 5 bidirectional levels with 500 \( \tanh \) units in each, giving it approximately the same number of weights as the DBLSTM networks. Retraining with weight noise was not found to be effective for the DBRNN, and the results are only quoted without noise. The best result of 17.99 \( \pm \) 0.13 is not significantly different from the best transducer result, which is the best TIMIT result we know of in the literature. The DBLSTM result without weight noise is better than the CTC result without noise, and the DBRNN hybrid result is much better than the DBRNN CTC result of 37.6 quoted in \([1]\).

### 5. WALL STREET JOURNAL EXPERIMENTS

The second set of experiments were carried out on the Wall Street Journal (WSJ) speech corpus. Their main purpose was to gauge the suitability of hybrid DBLSTM-HMM for large vocabulary speech recognition, and in particular to compare the approach with existing deep network and GMM benchmarks.

We trained an sGMM-HMM baseline system on WSJ corpus (available as LDC corpus LDC93S5B and LDC94S13B) using Kaldi recipe s5 \([20]\). The training set used for the experiments was the 14-hour subset train-si84, rather than the full 81 hour set. We used the dataset test-dev93 as the development set. The audio data was preprocessed into 40 dimensional log mel filter-banks, with deltas and accelerations, as with TIMIT. The trigram language model used for the task was provided with the WSJ CD. The forced alignments were generated from Kaldi recipe tri4b, corresponding to LDA preprocessing of data, with MMLT and SAT for adaptation. See Kaldi recipe s5 for further details. There were a total 3385 triphone states in the alignments.

The DBLSTM network had five bidirectional hidden levels, with 500 LSTM cells in each of the forward and backward layers, and a size 3385 softmax output layer, giving a total of 29.9M weights. The training parameters for the DBLSTM network were identical to those used for TIMIT. The deep network (DNN) had a context window of 15 acoustic frames (seven to either side of the centre frame being classified) It had six hidden layers with 2000 sigmoidal units in each, and a size 3385 softmax output layer. The DNN weights were
Table 3. WSJ Results. All results recorded on the dev93 evaluation set. ‘WER’ is word error rate, ‘FER’ is frame error rate and ‘CE’ is cross entropy error in nats per frame.

<table>
<thead>
<tr>
<th>System</th>
<th>WER</th>
<th>FER</th>
<th>CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLSTM</td>
<td>11.7</td>
<td>30.0</td>
<td>1.15</td>
</tr>
<tr>
<td>DBLSTM (NOISE)</td>
<td>12.0</td>
<td>28.2</td>
<td>1.12</td>
</tr>
<tr>
<td>DNN</td>
<td>12.3</td>
<td>44.6</td>
<td>1.68</td>
</tr>
<tr>
<td>sGMM [29]</td>
<td>13.1</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

initialized with samples from a mean of 0, standard deviation 0.067 Gaussian. The DNN was trained with stochastic gradient descent, starting with a learning rate of 0.1, and momentum of 0.9. The learning rate was reduced by a factor of 2 at the end of each epoch which failed to produce an improved WER over the previous epoch, on the development set. After six failed attempts, the learning rate was deemed to be low enough that no further annealing was performed. The network was trained for a total of 30 epochs. The posterior probabilities returned by the DNN and DBLSTM were not divided by state priors during decoding.

The DBLSTM networks outperformed both the GMM baseline and the DNN. However the improvement in word error over the DNN was much smaller than the gains in cross entropy error and frame error rate. Furthermore, retraining DBLSTM with noise decreased the cross entropy and FER, but increased the WER. The DNN was not pretrained as a Deep Belief Network [6], which may have considerably improved its performance.

6. DISCUSSION

The two sets of experiments in this paper deliver a mixed message about DBLSTM-HMM hybrids. On one hand the TIMIT results show that they can deliver results on a par with end-to-end discriminatively trained DBLSTM, and substantially better than the best deep networks. On the other hand, the Wall Street Journal results suggest that DBLSTM does not perform much better than a randomly initialised deep network, when used as an acoustic model in a hybrid system for large vocabulary speech recognition.

The picture is further complicated by the fact that DBLSTM is much better than deep networks at optimising the objective function used for training (cross entropy error). This is not entirely surprising, since DBLSTM is able to make more use of surrounding context than deep networks, and also able to ensure that it makes similar predictions for consecutive frames (unlike deep networks, where each prediction is made independently). It is therefore difficult to know how to improve the performance of DBLSTM within a hybrid system. Indeed adding a regularisation term reduced the Wall Street Journal cross entropy error still further, but made the word error rate worse.

The fundamental problem is that the frame-level distributions the network is trained to optimise are significantly different from the sequence level distribution that is implicitly defined by the decoding lattice. First of all, the network is trained to model the location of the state-segment boundaries, which may be inaccurate (if the forced alignment used to define them is suboptimal) and which is anyway irrelevant to the word-level distributions defined by the lattice (because the lattice sums over all feasible alignments). Secondly, the cross entropy error does not take into account the language model.

We suspect that the latter effect was the main reason for the disparity between cross entropy and word error for DBLSTM on Wall Street Journal. Given that DBLSTM can access context from across the whole sequence, it may be able to learn an implicit word-level language model from the training data which then interferes with the explicit language model used during decoding. For a task like TIMIT, with a weak, biphone language model estimated from the training utterances, the interference is likely to be negligible. For Wall Street Journal, where the language model is trained on a large text corpus and has a profound impact on overall performance, it may be more significant.

7. CONCLUSIONS AND FUTURE WORK

We have applied a hybrid HMM-Deep Bidirectional LSTM system to the TIMIT and Wall Street Journal speech databases. We found that the system gave state-of-the-art results on TIMIT, and outperformed GMM and deep network benchmarks on WSJ.

In the future we would like to apply the system to large vocabulary recognition on corpora with spontaneous speech, such as Switchboard, where the language model is likely to play a less significant role. We would also like to investigate full sequence training as a means of reducing the disparity between the objective function used for training and the performance during decoding.

8. REFERENCES


