Bengali Speech Recognition: A Double Layered LSTM-RNN Approach

Md Mahadi Hasan Nahid
Computer Science and Engineering
Shahjalal University of
Science & Technology
Sylhet-3114, Bangladesh
Email: nahid.cse.sust@gmail.com

Bishwajit Purkaystha
Computer Science and Engineering
Shahjalal University of
Science & Technology
Sylhet-3114, Bangladesh
Email: iambishwa@student.sust.edu

Md Saiful Islam
Computer Science and Engineering
Shahjalal University of
Science & Technology
Sylhet-3114, Bangladesh
Email: saiful-cse@sust.edu

***Abstract*—Speech recognition may be an intuitive process for humans, but it turns out to be intimidating to make computer automatically recognize speeches. Although recent progresses in speech recognition have been very promising in other languages, Bengali lacks such progress. There are very little research works published for Bengali speech recognizer. In this paper, we have investigated long short term memory (LSTM), a recurrent neural network, approach to recognize individual Bengali words. We divided each word into a number of frames each containing 13 mel-frequency cepstral coefficients (MFCC), providing us with a useful set of distinctive features. We trained a deep LSTM model with the frames to recognize the most plausible phonemes. The final layer of our deep model is a softmax layer having equal number of units to the number of phonemes. We picked the most probable phonemes for each time frame. Finally, we passed these phonemes through a filter where we got individual words as the output. Our system achieves word detection error rate 13.2% and phoneme detection error rate 28.7% on Bangla-Real-Number audio dataset.**

***Index Terms*—Bengali speech recognition, MFCC, RNN, LSTM, Phonemes, ASR.**

I. Introduction

Automatic speech recognition (ASR) ability of machines reduces the communication complexity between humans and them. Written instructions have been the primary way to communicate with machines. Since the last decade there has been a tremendous resurgence of research works in the field of ASR [1]. The models for ASR have been consistent in reducing the error rate while recognizing the speeches. Although quality of speech recognition in other languages is increasing day by day, Bengali speech recognition has drawn a very little interest [2]. This is likely due to the fact that Bengali words are very complex and there's a paucity of comprehensive data to train any model. We propose a deep long short term memory (LSTM), a special recurrent neural network, for individual Bengali word recognition.

ASR is different than other sequence learning problems in the aspect that speeches are variable-length. The same word can be spoken in multiple durations. The traditional approach of training a model for recognition has been, mostly, hidden markov model (HMM) [3], [4], [5]. HMM was popularized by the fact that it can tackle variable lengths by incorporating dynamic time warping (DTW) [6]. Generally, a small set of phonemes comprise all the words in languages. This is also exploited by HMM; they model the acoustic features and small set of phonemes well.

Nevertheless, the traditional model suffers a practical issue with speech recognition; it assumes that observations are identically and independently distributed. But this is clearly not true for the speeches. To overcome this limitation Gaussian mixture model (GMM) has been combined to HMM, and this turned out to be effective [7]. But recently it was revealed that the hybrid models of deep neural networks with HMM produce better recognition rate by very good margin [8], [8]. It turns out that the deep networks approximate better posterior probability than the GMM [9]. Hybrid model of combining several other models (e.g, deep model, HMM model, and singular value decomposition) is also very effective [10].

Parallely, the recurrent neural networks are exceptionally good with learning audio sequences [11], [12], [13]. They have recurring connections to store memory. This is a direct advantage over the HMM. LSTM, a popular recurrent network, stores and retrieves memory by using gates [14]. With LSTM, the phons (phonemes) in individual words can be detected with higher accuracy and equal efficiency [15]. LSTM can be convolved too; it can achieve better results in terms of *word error rate* (WER) for end-to-end ASR models [16]. As there are ample evidences that deep neural networks can model speeches efficiently in other languages, they are likely to perform well in Bengali speeches also.

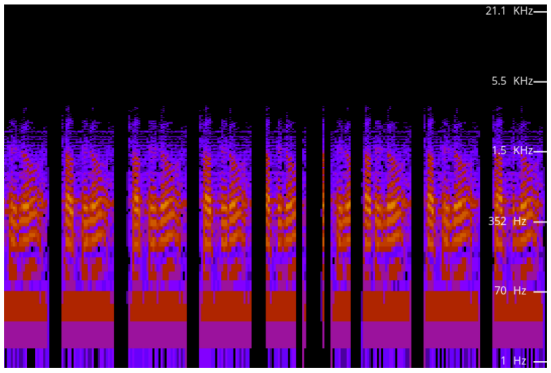
II. Methodology

A. Extracting Features from Signals: MFCC

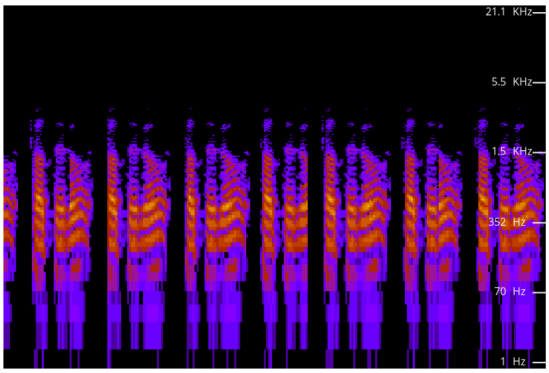
The raw audio speeches are less fit for direct use. The signals carry many redundant information along with noise that can plague the recognition task severely. Figure 1 shows spectra of the same audio signal before and after noise reduction respectively. It is, then, necessary that only the highly distinctive features from the signals be extracted. We have taken the nonlinear coefficients from mel-frequency cepstrum (MFC) from the audio speeches. The coefficients carry highly distinctive features (that are useful for discrimination) of the signal, collectively known as the mel-frequency cepstral coefficients (MFCC). The noise reduction prior to calculating

978-1-5386-1150-0/17/$31.00 2017 IEEE

MFCC is substantial. We have taken 13 coefficients for each time frame in the audio speeches.



Spectrum before noise reduction



Spectrum after noise reduction

Fig. 1: Effect of noise in audio signals

We divided each audio example into a number of frames. The number of frames for an example depended on the duration of that audio sample. For each frame $x_i(n)$ of the original audio signal $x(n)$ we calculated the complex valued discrete Fourier transform (DFT), where i denotes the frame index. For each $x_i(n)$ we had a corresponding $\chi_i(k)$ in frequency domain (Equation 1).

$$\chi_i(k) = \sum_{n=1}^{N} x_i(n)h(n)e^{-j\frac{2\pi}{N}kn} \quad (1)$$

Here, $h(n)$ is the hamming distance, and k is the length of DFT. Having calculated the DFT, estimation of power spectrum $P_i(k)$ for each individual frame i is straightforward.

$$P_i(k) = \frac{1}{N}|\chi_i(k)|^2 \quad (2)$$

After taking the square of the complex Fourier transform we kept only 257 coefficients, which were in turn, multiplied by 26 trianglular filters; then we added the coefficients which produced 26 numbers. Discrete cosine transform (DCT) was applied to the log of these numbers, and finally we got 26 ceptral coefficients. Out of those 26 coefficients we only kept 13 coefficients as the feature set for each frame. Figure 2 shows the MFCC for the same noise reduced audio signal of Figure 1.

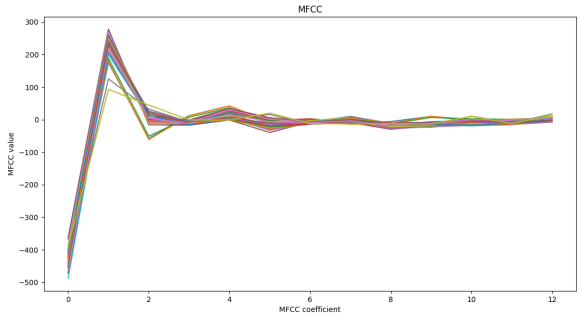


Fig. 2: MFCC values of 13 coefficients

Bengali	English	Bengali	English	Bengali	English	Bengali	English
অ	O	ক	K-O	ঠ	Tth-O	ভ	Bh-O
আ	A	খ	Kh-O	ড	Dd-O	ম	M-O
ই	I	গ	G-O	ঢ	Ddh-O	য	Z-O
ঈ	I	ঘ	Gh-O	ণ	N-O	র	R-O
উ	U	ঙ, ং	Ng	ত,ৎ	T-O	ল	L-O
ঊ	U	চ	C-O	থ	Th-O	শ	Sh-O
ঋ	R-I	ছ	Ch-O	দ	D-O	স	S-O
এ	E	জ	J-O	ধ	Dh-O	হ	H-O
ঐ	O-I	ঝ	Jh-O	প	P-O	ড়	R-O
ও	O	ঞ	N-I-O	ফ	Ph-O	ঢ়	R-O
ঔ	O-U	ট	Tt-O	ব	B-O	য়	Y-O

Fig. 3: Phon-based representation of Bengali alphabets in English

B. Phonemes Mapping and Labeling

Bengali has got a wide range of phonemes (phons). The pronunciation order of the phons within individual words in Bengali does not conform to the spelling like the other languages like English. For example, the word ‘play’ will be pronounced as /p/l/a/y/. The pronunciation order follows the text, ‘/l/’ will never be pronounced before ‘/p/’. To contrast in most cases Bengali word spelling does not conform to the ordering of how individual phons are pronounced. For example, Bengali word ‘খেলা’ is pronounced as /kh/e/l/a/ but in text ‘/e/’ comes before ‘/kh/’. Therefore, we labeled each data example in English letters. This ensured the ordering of pronunciation of individual phonemes conformed with their order in the text. Figure 3 shows the mapping of how individual Bengali letters are mapped to phone based representation in English.

The audio examples were dichotomized into frames containing 13 features each. Each frame was needed to be assigned a phon. We assumed that the phons within the individual examples will occupy equal number of frames, although this is not correct. In practice, some phons are pronounced with more duration than others and this also depends on the individual speakers. Moreover a good number of frames will contain overlapping phons. Therefore, it can’t be said beforehand which frame would contain what phon. But the idea is to introduce the correct phons for the frames to our LSTM network to the extent possible. We, therefore, didn’t distribute

TABLE I: Labeling the data examples

Speech	Phones	# Frames	Label
পঞ্চাশ	P-O-N-C-A-Sh	19	PPPP-OO-NNN-CCC-A ShShShSh
সতের	S-O-T-E-R-O	17	SSS-OO-TTT-EEEE-RR OO
বাইশ	B-A-I-Sh	13	BBB-AAAA-III-ShShSh
পয়ষট্টি	P-O-Y-Sh-O-Tt-Tt-I	26	PPPP-OOOOO-YY-ShS OOOO-TtTtTt-TtTtTt-II
কোটি	K-O-Tt-I	8	KK-OO-TtTt-II
দশমিক	D-O-Sh-O-M-I-K	18	DD-OOO-ShShShSh-O(II-KKK
হাজার	H-A-J-A-R	15	HHH-AAA-JJJJ-AAA-R
শূন্য	Sh-U-N-N-O	16	ShShShSh-UUUU-NNN NN-OOO

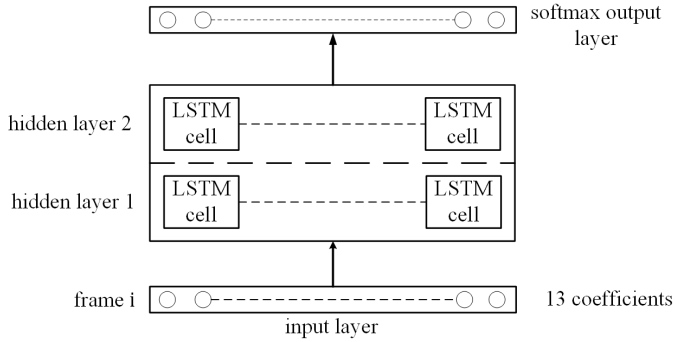


Fig. 4: LSTM recurrent neural network for speech recognition

the phons into equal number of frames, rather, we imposed random disparity (although in a very small amount) in the number of frames that they occupied. We did these random manipulation 3 times for each training examples so that it becomes more balanced. A set of few examples is shown in Table I.

C. Recognition: LSTM

Our model is a deep recurrent neural network with two layers of 100 LSTM cells each. Figure 4 shows our model. The bottommost layer is the input layer where we inject each time frame of an individual example at each time step. The layer contains 13 units that would contain the coefficients of the time frames. The next layer is two layers are LSTM recurrent layers. The final layer is a softmax output layer that contains 30 units as we have identified 30 individual phonemes. When examples are fed into the network in mini batches we get a probability distribution over each phons and we pick the phon with highest probability.

The interconnected LSTM cells are very efficient in modeling the sequences. Our LSTM cell is shown in Figure 5. Beneath the bottom edge of the cell, x_t is the 13 dimensional input frame at timestep t. The LSTM cell has a set of very selective gates which makes them so powerful in selectively remembering things. The cell has a *forget gate* which decides what to remember and what not to. The gate is built by

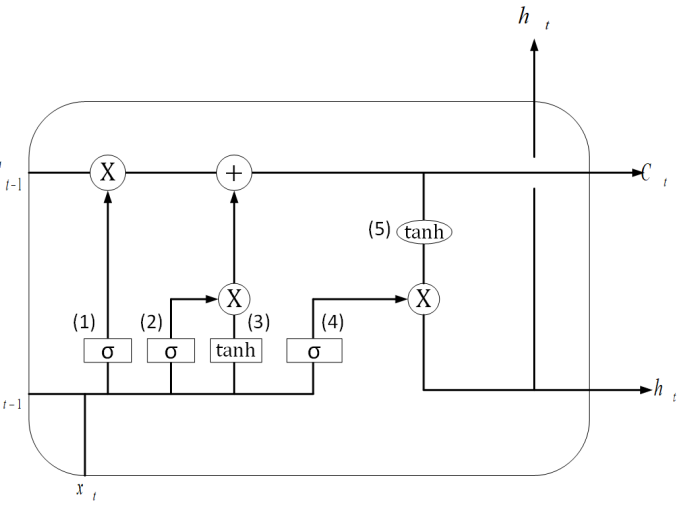


Fig. 5: An LSTM cell

combining the input with hidden state of previous time step, and applying a logistic activation on them (gate (1) in Figure 5).

$$f_t = \sigma\big(w_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{3}$$

Here $\sigma(\cdot)$ is a logistic activation function where $\sigma(x) = \frac{1}{1+e^{-x}}$. This gate is necessary as it possibly can store the information about correlation among phons. If the coefficients of input x_t cannot provide sufficient evidences of any phon, then this gate might tend to perceive that the current frame t contains the same phon of frame $t-1$. On the other hand, if a frame t provides substantial evidence of a different phon from frame $t-1$, then it might tend to forget the phon of $t-1$.

Gate (2), the *input gate*, decides what values to update. Based on previous time step's hidden state and the input, the *cell state* C_t (Equation 4) is updated after a series of operations.

$$\begin{aligned} i_t &= \sigma\big(w_i \cdot [h_{t-1}, x_t] + b_i\big) \\ c_t &= \tanh(w_c \cdot [h_{t-1}, x_t] + b_c) \\ C_t &= f_t \odot C_{t-1} + i_t \odot c_t \end{aligned} \tag{4}$$

The output of a cell state is calculated in gate (4), and it is then used for calculating the hidden state at t which would be required for following time step.

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{6}$$

All the outputs of the second LSTM layer are then into a softmax layer for the discrimination task. The objective is to minimize the cross entropy loss function of the training speeches. Both the targets (t_i) and outputs(y_i) of each frame are one-hot encoded. Therefore, we try to minimize L at each iteration of training.

$$L = -\sum_i t_i \log y_i \tag{7}$$

D. Post Processing

The raw output produced by our network is not directly comparable. At first we stripped out the phons that are surely noisy. Then we had to filter the output to produce convincing outputs so that this could be compared to the actual word that the audio speech had contained. For each output string we filtered it using the procedure described in Algorithm 1. As it is extremely rare for any text to have more than two consecutive same letters we restricted detecting the phons upto two consecutive places. It turns out that the filter eliminates noisy phons substantially well.

Algorithm 1 Filtering the output string procedure

$S \leftarrow$ output string
$S_{flt} \leftarrow$ “”
$len \leftarrow$ end of S
$i \leftarrow$ start of S
$T \leftarrow$ threshold applied to s
FIRST:
  if $i > len$ **then**
    return S_{flt}
  end if
  $j \leftarrow$ i + 1
  $c \leftarrow 1$
SECOND:
  if $j \leq len$ **and** $S_i = S_j$ **then**
    $c \leftarrow c + 1$
    $j \leftarrow j + 1$
    goto SECOND
  end if
$times \leftarrow T \times c/len$
if $times \geq 1.8$ **then**
  $S_{flt} \leftarrow S_{flt} + S_i + S_i$
else if $times \geq 0.8$ **then**
  $S_{flt} \leftarrow S_{flt} + S_i$
end if
$i \leftarrow j$
goto FIRST

After filtering, we looked for most plausible words in our vocabulary to match the phonemes. The matched words were then decided to be final outputs of our model. The value for the threshold T as defined in the algorithm required to be set empirically which we describe in Section III-C.

III. Experimental Results and Analysis

A. Dataset

We have tested our system on a dataset that contains total 2000 words ([2]). In short, there are total 114 unique words in the dataset. The audio speeches were contributed by 15 different male speakers of age in 20-24. The dataset contained only Bengali real numbers. We splitted the dataset into 75:12.5:12.5 for training, validation, and testing purposes respectively. Training set comprised of approximately 37.5 minutes audio samples whereas validation and test sets were about 6.25 minutes each.

Fig. 6: Error rates vs number of hours being trained

B. Validation Error rates

We traced two kinds of error made by our network. The individual *phon detection error rate* E_p; it is the mismatch rate between the raw outputs of our model and the spurious we assigned to validation examples as in Table I.

$$E_p = \frac{\#\text{ mismatches in phons}}{\#\text{ mismatches in phons} + \#\text{ matches in phons}} \times 100\% \tag{8}$$

Concurrently, the other kind of error was measured only after post-processing the raw output, *word detection error rate* E_w. It is calculated in terms of number of mismatches between output words and the actual words contained in the validation examples.

$$E_w = \frac{\#\text{ mismatches in words}}{\#\text{ mismatches in words} + \#\text{ matches in words}} \times 100\% \tag{9}$$

Figure 6 shows how our model was learning with respect to the number of hours it was being trained. The system operated on Linux with core-i7 processor and 8 GB memory. The hourly plotted curves make sense. At the beginning of training individual phons were recognized more frequently than that for words. This is due to the fact that, although relevant phons were detected but they were inconsistent; when they were passed for filtering the model couldn't pick suitable words. After few iterations the red E_w curve was mostly aided by the post-processing filtering step and this was another reason why the E_w curve undershot the blue E_p curve. Moreover, the frames were not labeled loosely as we had mentioned in Section II-B. This is another reason why phon detection error rate did not descend well as compared to word detection error rate.

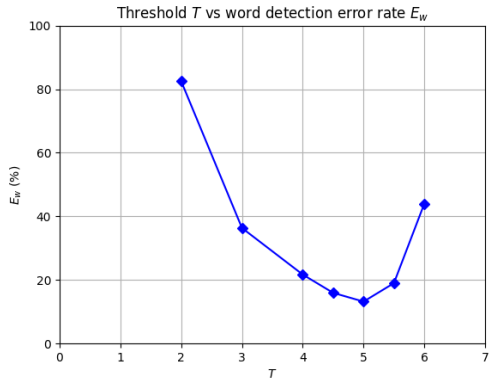


Fig. 7: Impact of threshold T on E_w

C. Threshold Setting

The threshold value T of post-processing has a great impact on the word detection error rate E_w. It actually reflects the nature of dataset. If the dataset contains words that are of mostly short length then smaller value of T would produce better results and vice versa. We got smallest E_w when the value of T was set to 5. This is shown in Figure 7.

D. Final Test Error and Output Analysis

The test run produced 28.7% phon detection error rate and 13.2% word detection error rate. We have analyzed on what words our system fails most. The analysis revealed that our system, many times, garbles one phon with other phon that is uttered similarly (see Figure 3).

T $\longleftrightarrow$ Tt	Tt $\longleftrightarrow$ Tth	B $\longleftrightarrow$ Bh
G $\longleftrightarrow$ Gh	D $\longleftrightarrow$ Dh	D $\longleftrightarrow$ Dd

On the other hand, some phons are almost always detected with higher confidence such as 'A', 'O', 'L', etc. These phons occured most number times and does not have any other phon uttered similarly. Matching the output of the network with vocabulary causes our model not to recognize any word outside its vocabulary. An example is shown below where an audio signal containing the word 'ষোল' (Sh-O-L-O) is fed into the system. The system calculates the MFCC of the signal. Say, it has 12 time frames.

Audio speech	$\longrightarrow$	Sh-O-L-O
Labeled	$\longrightarrow$	`Sh,Sh,Sh,O,O,O,L,L,O,O,O,O`
Raw output	$\longrightarrow$	`S,Sh,Sh,O,O,O,L,L,L,O,O,O`
Initial noise elimination	$\longrightarrow$	`_,Sh,Sh,O,O,O,L,L,L,O,O,O`

Filtering :
- 'Sh' $\leftarrow \frac{2}{\frac{3}{12}} \times 5 = 0.83$ (1 times)
- 'O' $\leftarrow \frac{3}{\frac{3}{12}} \times 5 = 1.25$ (1 times)
- 'L' $\leftarrow \frac{3}{\frac{3}{12}} \times 5 = 1.25$ (1 times)
- 'O' $\leftarrow \frac{3}{\frac{3}{12}} \times 5 = 1.25$ (1 times)

Although, the network produced one noisy phon (i.e., 'S') it has been eliminated by the system. Then the filter (Algorithm 1) applied threshold to each of the remaining consecutive phons and constructed the word 'Sh-O-L-O'. The constructed word will be looked up in the vocabulary and the word with smallest edit distance (with the newly constructed word) would be returned. Assuming that the given word exists in the dictionary, the model would output the correct word.

IV. Conclusion

The deep LSTM network is able to model Bengali speeches effectively. The phoneme-based speech recognition approach requires that the context of phons be taken into consideration while modeling. The recurrent architecture models the phonemes efficiently because it captures the context by using memory. We have solely emphasized on detecting individual Bengali words, but there are improvements possible with a larger dataset. The value set for the threshold at post processing is an ad-hoc procedure on which a further investigation can be made. The further improvement is possible with recognizing Bengali speeches at sentence level with an increased vocabulary.

References

[1] D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach*. Springer, 2014.

[2] M. M. H. Nahid, M. A. Islam, and M. S. Islam, "A noble approach for recognizing bangla real number automatically using cmu sphinx4," in *Informatics, Electronics and Vision (ICIEV), 2016 5th International Conference on*. IEEE, 2016, pp. 844–849.

[3] M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[4] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, "Hmm adaptation using vector taylor series for noisy speech recognition," in *Sixth International Conference on Spoken Language Processing*, 2000.

[5] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov models for speech recognition*. Edinburgh university press Edinburgh, 1990, vol. 2004.

[6] C. S. Myers and L. R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected-word recognition," *Bell Labs Technical Journal*, vol. 60, no. 7, pp. 1389–1409, 1981.

[7] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7398–7402.

[8] S. Ravuri and S. Wegmann, "How neural network features and depth modify statistical properties of hmm acoustic models," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5080–5084.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] S. Xue, H. Jiang, L. Dai, and Q. Liu, "Speaker adaptation of hybrid nn/hmm model for speech recognition based on singular value decomposition," *Journal of Signal Processing Systems*, vol. 82, no. 2, pp. 175–185, 2016.

[11] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[12] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.

[13] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The microsoft 2016 conversational speech recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5255–5259.

[14] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[16] Y. Ning, Z. Wu, R. Li, J. Jia, M. Xu, H. Meng, and L. Cai, "Learning cross-lingual knowledge with multilingual blstm for emphasis detection with limited training data," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5615–5619.