End-to-End Bengali Speech Recognition using DeepSpeech

M. M. H. Nahid*, B. Purkaystha, M. S. Islam

Department of Computer Science and Engineering
Shahjalal University of Science & Technology, Sylhet-3114, Bangladesh
E-mail: nahid-cse@sust.edu, iambishwa@student.sust.edu, saiful-cse@sust.edu

* Corresponding author [Md Mahadi Hasan Nahid]

Keywords:

- Bengali Speech Recognition;
- Bengali ASR;
- DeepSpeech;
- LSTM;
- CTC;
- GRU.

Abstract: One of the main challenges in Bengali speech recognition is modeling phoneme alignment. The exact prior probabilities of phoneme transitions in an individual word are not known; in this paper, we seek to investigate the DeepSpeech network for recognizing individual Bengali speech samples. There are recurrent LSTM layers at the heart of this network for modeling internal phoneme representation. We have added convolutional layers at the bottom which obviates the need of assuming anything about internal phoneme alignment. We have trained the model with a connectionist temporal classification (CTC) loss function and constructed the transcript by using a beam search decoder. We have tested our method on Bengali real number speech dataset; it incurs 8.20%-word error rate and 3.00%-character error rate on this dataset. The results significantly outperform current existing methods on this dataset.

1. Introduction

Type Speech recognition is the ability to comprehend the continuous stream of spoken words or sentences which may have different accents, occasional distortions, and unnecessary noises. Humans do not have to explicitly learn how to recognize speeches; they develop this ability naturally as they learn the language itself. In contrast, automatic speech recognition (ASR) refers to training the machines with a set of very sophisticated and carefully designed algorithms and to improving their accuracy gradually in detecting individual characters, words, and sentences. The ability of recognizing speeches automatically has several applications.

‘The most widely used and reliable interface between the humans and the machines has been through typing commands; the users write command or instructions and the machines follow or execute it. More often than not, typing or writing individual commands is tedious, and therefore, with automatic speech recognition the efficiency of the interaction between humans and computers greatly improves. It is easier to ask the Google about the weather than to write a search query. However, the automatic speech recognition has not been very successful until very recently. This is due to the fact that manual feature detection in individual speech samples does not contribute to higher accuracy in recognition, and for years, the computational resources have not been compatible with some methods which were able to detect useful distinctive features automatically. These methods were

Submitted: 08 May 2019 | Revised: 01 July, 2019 | Accepted: 08 July, 2019

mostly deep artificial neural networks which are proven to be the universal approximators (Hornik, K, 1991). As these methods required huge amounts of data and were very slow to train, the early success on speech recognition depended on the other methods, such as hidden markov model (HMM) (Acero, A et. al., 2000; Gales, M.J, et al., 1998).

Most hidden markov models maintain a set of internal hidden states by which they seek to model individual phonemes over given period of time frames, and finally, they gather these individual phonemes together to construct words and sentences (Nwe, T.L., et al., 2003). Gaussian mixture model (GMM) mitigates one of the practical issues of the HMM (which is, it assumes that observations are identically and independently distributed) by combining with HMM in which GMM parameters are learned along with HMM parameters to approximate the prior bias in observations (Seltzer, M.L., et al., 2013). To put in other words, the transition from one phoneme to another phoneme is not totally independent and there is a prior probability for the transition; the GMM seeks to learn these probabilities.

However, as the expectations from ASR was increasing, these probabilistic methods were not able to scale up; concurrently, however, the computation power of the machines were increasing very rapidly, and deep neural networks were once again approached. With the increased computational power, the neural network produced very promising results (Hinton, G., et al., 2012; Yi, J., et al., 2018). The recurrent connections within the individual layers helped them model the temporal influence and the prior bias in observations. The improvements made by these deep neural network architectures are significant in English (Zhang, Y., et al., 2017) in Mandarin (Hannun, A., et al., 2014), in Hindi and other languages (Dua, M., et al., 2018). Unexpectedly, Bengali automatic speech recognition has not seen such improvement over the years (Swarna, S.T., et al., 2017).

Hidden markov model (HMM) using CMU Sphinx 4 (a popular speech recognizer from Carnegie Mellon University) has been used to recognize Bengali real numbers (Nahid, M.M.H., et al., 2016). It has a few limitations; the model does not generalize well with the individual speakers, that is, it is highly sensitive to pitch, intonation, and accent. A deep unidirectional long short-term memory (LSTM) neural network was proposed to address these issues (Nahid, M.M.H., et al., 2017). The authors first tried to approximate the phone alignment in individual speech samples using a statistical approach to build their input set. They, then, fed these input samples to a two-layer unidirectional LSTM network and trained their model. They succeeded to achieve higher accuracy in terms of recognition; however, they were recognizing individual words instead of full sentences. This limitation may be ascribed to their statistical assumption in phone alignment in individual speech samples.

In this work, we provided an improvement to the phone assumption alignment using connectionist temporal classification (CTC) loss function (Graves, A., et al., 2006). Our work is strongly inspired by an end-to-end deep speech recognition method (Amodei, D., et al., 2016, Hannun, A., et al., 2014). The method has been very successful in English and Mandarin languages, and meddling the method had the potential for recognizing Bengali speech samples without assuming anything about the internal phone alignment. We had a recurrent long short-term deep architecture similarly to (Nahid, M.M.H., et al., 2017); however, our architecture is bidirectional which means it should model the prior transition probabilities better than a unidirectional model would do. The

Email: nahid-cse@sust.edu

bottommost two layers are convolutional layers; these two layers are feature detectors in the fourier spectrum. They relay the features in individual speech samples to upper recurrent layers which are in turn connected to densely connected feedforward layers. The final layer of the network is a softmax layer which outputs a probability distribution. Some improvements are achieved in end-to-end speech recognition using CNN-RNN-CTC based model on noisy data and gained 10.65 % Character Error Rate (CER) (Sumit, S.H et al., 2018).

In our work, we make several contributions. We remove the barrier in assuming phone alignment in speech samples which ultimately translates into a better accuracy (see Section 3.5). The previous unidirectional two-layer LSTM architecture was only good at recognizing individual words; in our work, we recognize speech samples in a unit of sentences without compromising the accuracy. In fact, the accuracy is significantly higher even when individual sentences are recognized (see Table 2).

The rest of the paper is organized as follows. Section 2 describes our method in detail. Section 3 follows with our findings in different experiments which we have conducted and shows the comparative analyses of our approach with other existing methods (see Table 2). Finally, in Section 4, we summarize our findings and briefly discuss about future extensibility.

.

2. Methodology

The method can be divided into three serialized stages. In the first stage, we need to preprocess the raw speech samples, the following step is the training for actual recognition task, and the final step is to decode the output of the recognizer network to form meaningful transcriptions.

2.1 Preprocessing

The preprocessing step is necessary because raw speech samples are seldom useful for the task. Therefore, we had to reduce the unnecessary noises, random distortions at first; we, then, applied Fourier transformation to the individual audio samples and finally extracted highly 13 distinctive nonlinear mel frequency cepstral coefficients (MFCCs) from them. These coefficients are very popular in the literature for recognition task as they are highly discriminative across phones.

The speech samples are divided into a number of frames (possibly different number of frames for different samples depending on the duration). Say for an audio signal $x(n)$ is divided into a number of frames and a frame is denoted by $x_i\,(n)$ having the frame index i. We calculated discrete Fourier transform (DFT) and for each $x_i\,(n)$ we had a corresponding $X_i(k)$ in the frequency domain (see Equation 1).

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

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

In Equation 1, $h(n)$ is the hamming distance whereas k is the length of DFT. The power spectrum $P_i(k)$ for an individual frame i can be effectively calculated as in Equation 2. We picked 257 most significant coefficients and multiplied them with 26 triangular filters; then we added the coefficients to produce 26 numbers. Discrete cosine transform (DCT) was applied to the log of these numbers, and finally 13 cepstral coefficients out of 26 were kept for each frame. Therefore, for a speech sample which had 20 frames, after preprocessing, we represented it in a 20 × 13 vector.

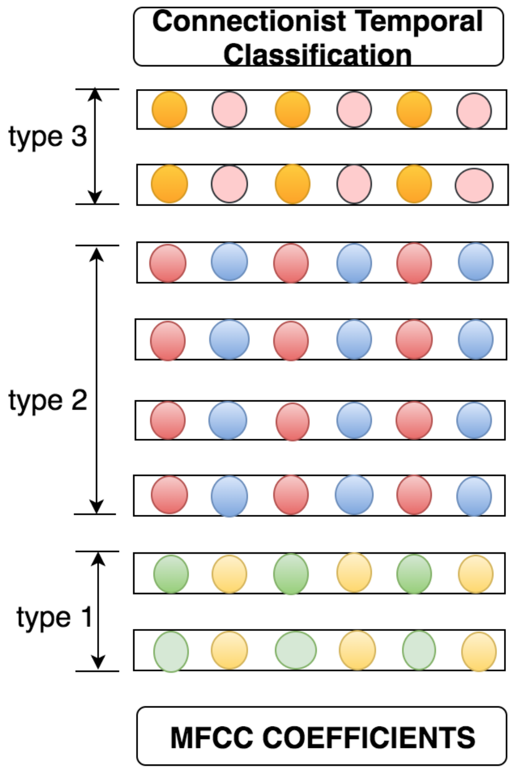

Fig. 1 Deep speech architecture contains three different types of serialized layers. The bottommost layers are convolutional layers, the layers immediately above it are recurrent layers, and the uppermost layers are the fully connected layers.

2.2 Recognition Network: Bidirectional LSTM with CTC Loss Function

Our recognition network is the one of (Amodei, D., et al 2016) at the core. The full architecture is shown in Figure 1 and there are actually three types of units (or layers in combination) in this deep network. The input to the network is power spectra what we get after the preprocessing step. The network is trained with connectionist temporal classification (CTC) loss function.

The bottommost layer connects to the cepstral coefficients and adjusts its weights to detect useful features. The bottommost layers (i.e., type-1 layers) are actually convolutional layers. The inputs are fed into the convolutional layers in mini batches. However, it is very unlikely for all the samples in a minibatch to have the same number of frames; this is why padding frames (which obviously contain void speech or sound) need to be appended to the samples whenever appropriate. As the input are fed into the network in timestamps, for a reference window of size *k*, the *i-th* unit in a convolutional layer l at a timestamp t outputs as in Equation 3.

$$h^{(l,i)} = \sigma\left(\omega^{(l,i)} \odot h_{t-k:t+k}^{l-1}\right) \qquad (3)$$

Here $h^{(l,i)}$ denotes the output of the *i-th* unit of layer *l*, and therefore by definition, $h^{(0)}$ denotes the input and it contains 13 units (that is, each unit represents exactly one coefficient). *σ(·)* is the type of activation function we use, and for our task, we have actually used a modified relu activation.

$$\sigma(x) = min(max(0, x), 5) \qquad (4)$$

We always constrict the output of a convolutional unit to be less than or equal to 5.

The convolutional layers are followed by type-2 layers which are actually recurrent bidirectional LSTM layers. At any timestamp *t*, the units at the layer *l* takes update both from past timestamp and future timestamp.

$$\overrightarrow{h_t^l} = tanh(\omega^l \cdot h_t^{l-1} + \overrightarrow{U^l} \cdot \overrightarrow{h_{t-1}^l} + b^l) \qquad (5)$$

$$\overleftarrow{h_t^l} = tanh(\omega^l \cdot h_t^{l-1} + \overleftarrow{U^l} \cdot \overleftarrow{h_{t+1}^l} + b^l) \qquad (6)$$

Email: nahid-cse@sust.edu

In Equation 5 and Equation 6, ω^l is the input-hidden weight matrix whereas U^l (i.e., both forward going and backward going) is the recurrent weight matrix, and finally b^l is the bias. Adding these forward directional and backward directional states yields to an "informed state" ($h^l = \underset{h^l}{\rightarrow} + \underset{h^l}{\leftarrow}$) which is shaped by the prior transitional probabilities of the phonemes. The activation function *tanh(·)* acts like a squashing function (as it forces the output to be in the range [-1,1], Equation 7)

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7}$$

The processed cepstral coefficients flow through the recurrent layers and each upper layer gets this processed information from its immediate lower layer (Equation 8).

$$h_t^l = f\left(\omega^l \cdot h_t^{l-1} + b^l\right) \tag{8}$$

The recurrent layers are followed by type-3 layers which are fully connected from the bottom to the top. The output is a softmax layer which gives a probability distribution over the phonemes (see Equation 9).

$$P\left(o_t^k = k|x\right) = \frac{e^{\omega_k^L \cdot h_t^{L-1}}}{\sum_i e^{\omega_k^L \cdot h_t^{L-1}}} \tag{9}$$

The value of the output unit o_n at any timestamp t will indicate the probability of the corresponding phoneme n as predicted by the network. The network is then trained using the CTC loss function (Graves, A., et al., 2006), and the parameters of the network (that is, the biases and weights) are updated using backpropagation through time (BPTT) algorithm.

We finally use a 32-bit beam search decoder to construct the output from the phoneme distribution.

3. Experiment and Results

3.1 Dataset and Error Matrix

We have tested our system on a dataset that contains Bengali real number speech samples (Nahid, M.M.H., *et al.*, 2018). The dataset contains more than two thousand speech samples which are mostly Bengali real numbers. There are total of 115 unique words which are distributed across these speech samples. The speech samples are fairly large, and on average each sample has around eight words in it. The dataset is not very comprehensive, meaning that, it only contains real numbers; most of the words that occur in daily conversations are absent in the dataset. Another limitation of the dataset is that it is significantly small in size for a comprehensive speech recognizer (that is, there is only a little above three and half hours of recording overall). The domain of the speeches compensates for its smaller size.

We have used two most popular error metrics of the speech recognition literature. These are word error rate (WER) and character error rate (CER). WER is defined as the edit distance between the two sentences (one is predicted by the method whereas the other one is the actual label of the sample). The edit distance is measured after the words are tokenized. Let $\boldsymbol{R_w}$ be the number of substitutions, $\boldsymbol{D_w}$ be the number of deletions, and $\boldsymbol{I_w}$ be the number of insertions of words required to make two samples identical. Let further assume that number of correct words that matched is $\boldsymbol{C_w}$ Then, word error rate (WER) is calculated as follows.

$$\boldsymbol{WER} = \frac{\boldsymbol{R_w + D_w + I_w}}{\boldsymbol{R_w + D_w + C_w}} \tag{10}$$

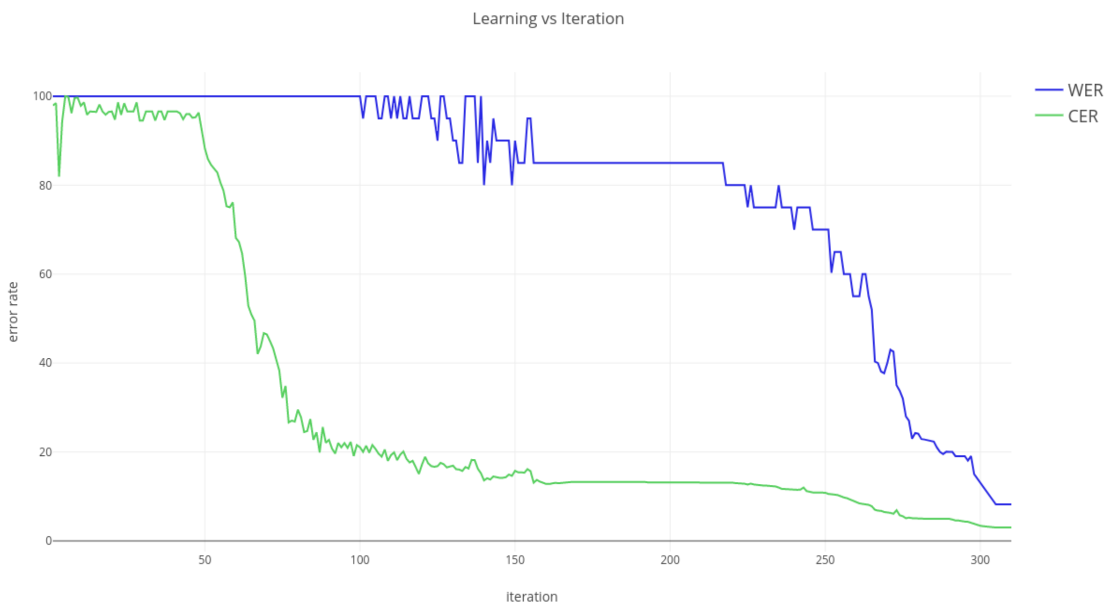

Fig. 2 Validation error rates plotted against iterations.

Very similarly, the character error rate (CER) is also calculated. If $\boldsymbol{R_c}$ is the number of substitutions, $\boldsymbol{D_c}$ is the number of deletions, and $\boldsymbol{I_c}$ is the number of insertions of characters required to make two sentences identical, then CER is calculated as follows.

$$\boldsymbol{CER} = \frac{\boldsymbol{R_C + D_c + I_c}}{\boldsymbol{R_c + D_c + C_c}} \tag{11}$$

Again, $\boldsymbol{C_c}$ denotes the correct matching in characters prior to the process.

3.2 Experimental Settings

We identified 30 individual phonemes in the speech samples across the dataset. Therefore, the output layer of the network (i.e., softmax layer) had 30 units; it means that at each time-step the network was outputting a probability distribution over the thirty phonemes. The machine on which we ran our system on contains an Intel core i7 processor with 32 gigabytes of memory. The learning behavior shown by our model is plotted in Figure 2. This figure is interesting because there are many plateaus for both word error rate and character error rate. We split our training and testing dataset in ratio of 70:30.

Table 1 Error rates with different recurrent units

Configuration	Basic RNN	GRU	LSTM
7 layers (3 recurrent)	(17.41, 6.53)	(10.32, 4.09)	(9.68, 4.07)
8 layers (5 recurrent)	(16.78, 6.28)	(9.06, 3.30)	(8.47, 3.19)
11 layers (7 recurrent)	(14.76, 5.01)	(8.65, 3.15)	**(8.20, 3.00)**
12 layers (8 recurrent)	(14.82, 5.01)	(8.90, 3.37)	(8.36, 3.10)

We had to train the network for 310 iterations to reach an acceptable local

Email: nahid-cse@sust.edu

optimum, and to arrive at this optimum, the system required five days of training on the aforementioned machine. Apparently, the learning of characters is consistent (e.g., look at the CER in iterations from 50 to 150). On the other hand, the word error rate does not decrease significantly as compared to the CER; but, once the CER goes down below a threshold, the descend in WER is faster.

3.3 Recurrent Units

We have replaced the LSTM units at the recurrent layers with other recurrent units. The results are given in Table 1. The first value in the pair denotes the word error rate whereas the second value in the pair denotes the character error rate. Optimal result was achieved with the LSTM cells when we used 7 recurrent layers. In each of the recurrent layers, we have used 768 recurrent units (for each configuration). Whenever we wanted to increase the capacity of our network, we introduced more layers instead of adding more recurrent units within a layer. With increasing number of layers, the model starts overfitting for both LSTM units and GRU units. On the other hand, the network with basic recurrent units is too naive to capture the regularity in the MFCC values even with an increasing number of parameters.

3.4 Impact of Batch Size

In this section, we discuss the impact of batch size on our model. In Figure 3, we have tried out different batch sizes (keeping everything other intact); the end result shows that we have incurred lowest WER and CER when we had set the batch size to 150. As for smaller batch sizes, the gradient received for the parameters tend to be more opinionated; we had to stop the training of the network because the validation error was zigzagging (although in a smaller parameter space). For larger batch sizes, the update received in the backpropagation tended to be averaged out by the opposite gradients and the validation loss converged to a local minimum early in the training. As for a physical interpretation, it is possible that the network tries to model the individual intonations, noise, and other accidental regularities when the batch size is small; in the process the training accumulates to overfitting and the validation loss, kind-of, wanders in the parameter space instead of decreasing. On the other hand, when the batch size is bigger, the network almost never gets to model the required regularities (e.g., implied prior probabilities in phone transitions) and training stucks in a local minimum.

3.5 Comparison with models

We compare our model with a few of the best models in the literature in this section. The authors in (Nahid, M.M.H., *et al.*, 2016) approached CMU Sphinx model which internally approximates the phoneme transitions by using the hidden markov model (HMM). Another method (Nahid, M.M.H., *et al.*, 2017) used a unidirectional two-layer LSTM network for recognizing individual phonemes; additionally, they proposed a new decoder that collects these phonemes and constructs individual words. As compared to the first model, it can only recognize individual words.

In Table 2, we have reported the error rates when the models were run in the same dataset. To calculate the character error rate of two-layer LSTM model, we had to modify the original decoder from the model.

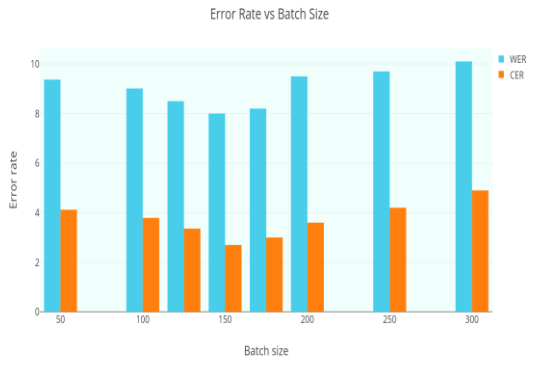

Fig. 3 Different batch sizes have different impact on the error rates. We achieved lower word error rate 3.00 and character error rate 8.20 when the batch size was set to 150.

We stopped the decoding process without calculating the edit distance with all the words and picking the word with the smallest edit distance. The authors had mentioned word detection error rate in their work, but for fair comparison, we reproduced the WER of their model. Therefore, it can be seen that our DeepSpeech model has produced lowest word error rate and character error rate as compared to the previous two models.

Table 2 Comparison across models with respect to error rates

Model	WER	CER
CMU Sphinx 4 model (Nahid, M.M.H., *et al.*, 2016)	15.00	-
Two layer LSTM model (Nahid, M.M.H., *et al.*, 2017)	23.78	37.49
DeepSpeech model	**8.20**	**3.00**

4. Discussions

To elaborate this finding, it should be mentioned that the recognizing portions of our DeepSpeech network and two-layer LSTM network are almost similar. Both use long short-term memory units at the core; but in this work, in addition, we have used bidirectional LSTM units. This helps in better modeling prior probabilities of phoneme transitions. For unidirectional units, the gradients are not received from future time-steps, and consequently, model lacks many information about phoneme transition regularities. Moreover, we have used seven layers of LSTM units which can capture more information as compared to two layers of LSTM units. In addition, the convolutional layers at the bottom detect useful features in the MFCC values in our model.

5. Conclusion

In this paper, we have proposed a DeepSpeech network for recognizing Bengali speech samples. The main contribution of our work is that it has produced lower word error rate and character error rate in Bengali literature as compared to the existing models. The statistical phoneme alignment was made unnecessary by the convolutional layers at the bottom and by the beam search decoder at the output. We have shown the performance measurement (in terms of error rates) for different recurrent architecture and it turned out that LSTM cells in the recurrent layers worked best for our scenario. We have also shown a comparative analysis with other best existing models in Bengali with respect to error rates. As a future task, we would like to increase the volume of the dataset by introducing more speakers; we will also increase our dictionary to encompass daily life conversational words to gauge true efficacy of the DeepSpeech network.

References

Acero, A., Deng, L., Kristjansson, T., Zhang, J.: Hmm adaptation using vector taylor series for noisy speech recognition. in Sixth International Conference on Spoken Language Processing (2000)

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. and Ng, A.Y., 2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al.: Deep speech 2: End-to-end speech recognition in english and mandarin. In: International Conference on Machine Learning, pp. 173–182 (2016).

Dua, M., Aggarwal, R., Biswas, M.: Discriminatively trained continuous hindi speech recognition system using interpolated recurrent neural network language modeling. Neural Computing and Applications pp. 1–9 (2018).

Gales, M.J., et al.: Maximum likelihood linear transformations for hmm-based speech recognition. Computer speech & language 12(2), 75–98 (1998).

Graves, A., Fern´andez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning, pp. 369–376. ACM (2006).

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al.: Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567 (2014).

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine 29(6), 82–97 (2012).

Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks 4(2), 251–257 (1991).

Nahid, M.M.H., Islam, M., Purkaystha, B., Islam, M.S., et al.: Comprehending real numbers: Development of bengali real number speech corpus. arXiv preprint arXiv:1803.10136 (2018).

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

Nahid, M.M.H., Purkaystha, B., Islam, M.S.: Bengali speech recognition: A double layered lstm-rnn approach. In: Computer and Information Technology (ICCIT), 2017 20th International Conference of, pp. 1–6. IEEE (2017).

Nwe, T.L., Foo, S.W., De Silva, L.C.: Speech emotion recognition using hidden markov models. Speech communication 41(4), 603–623 (2003).

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

Swarna, S.T., Ehsan, S., Islam, M., Jannat, M.E., et al.: A comprehensive survey on bengali phoneme recognition. arXiv preprint arXiv:1701.08156 (2017).

Yi, J., Wen, Z., Tao, J., Ni, H., Liu, B.: Ctc regularized model adaptation for improving lstm rnn based multi-accent mandarin speech recognition. Journal of Signal Processing Systems 90(7), 985–997 (2018).

Zhang, Y., Chan, W., Jaitly, N.: Very deep convolutional networks for end-to-end speech recognition. In: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, pp. 4845–4849. IEEE (2017).

Sumit, S.H., Al Muntasir, T., Zaman, M.A., Nandi, R.N. and Sourov, T., September. Noise robust end-to-end speech recognition for bangla language. In 2018 International Conference on Bangla Speech and Language Processing (ICBSLP) (pp. 1-5). IEEE (2018).