Neural Networks for Automatic Speech Recognition of Under-Resourced Languages: The Case of Amharic

Getachew Ahmed
HiLCoE, Computer Science Programme, Ethiopia
getabegaz@gmail.com

Solomon Teferra
HiLCoE, Ethiopia
Addis Ababa University, Addis Ababa, Ethiopia
solomon_tefera_7@yahoo.com

Abstract

This paper attempts to investigate the possibility of designing and developing a read-speech, large vocabulary and speaker independent speech recognition system for Amharic using neural networks. An Automatic Speech Recognition (ASR) system was designed using neural networks with the available computing resources. A probability-based connectionist temporal classification model was used to decode strings corresponding to an audio input. Five experiments were conducted using different hyper parameter tuning. Increase in the network size had an impact on the performance of the speech recognition system. A model was designed with early stopping regularization still with better performance. The best result was found with Recurrent Neural Network (RNN) architecture of 3-Gated Recurrent Unit (GRU) layers and 200 units per layer with early stopping regularization. Performance, based on character error rate for training, validation and testing sets, was 75.8%, 69.2% and 69.6%, respectively.

Keywords: Speech Recognition; Deep Learning; Neural Networks; Machine Learning

1. Introduction

Speech recognition is the mapping of an acoustic signal containing a spoken natural language utterance into the corresponding sequence of words intended by the speaker [1]. It is particularly relevant for Amharic which has more than 250 distinct consonant-vowel syllable characters [2] making writing on the keyboard difficult. A study by Ruan *et al.* [3] found that with speech recognition improves input rates than if one uses the keyboard for short message transcription.

Because of its wide range of applications, research and development in automatic speech recognition systems has recently emerged as one of the important developments in artificial intelligence. While there are some positive developments so far, Amharic speech recognition is still a challenging task. This is especially true given that Amharic is under-resourced and morphologically rich [2, 4]. There have been previous attempts on designing and developing Amharic speech recognition systems [3, 4, 5, 6, 7, 8]. Most of these studies were based on Hidden Markov Models (HMM) that model sequences of phonemes and Gaussian mixture models (GMM) which map the association between acoustic features and phonemes. N-gram model is also used for language modeling in these studies. Two exceptions are studies by Hussein Seid and Gamback [7] and Abebe Semie [9] who used a hybrid of HMM and artificial neural networks with some improvement in terms of performance as compared to the above mentioned works.

Literature shows that recurrent neural networks are the central components of state-of-the-art speech recognition systems [10]. ASR researchers in other languages are now developing an end-to-end speech recognition system where all the components of the HMM paradigm are replaced with some form of neural network architecture [11, 12, 13]. Research on ASR systems for Amharic using neural networks is, however, lacking in the literature. However, designing and developing ASR systems with neural networks entails bigger labeled speech data. Training of neural networks in general, and recurrent neural networks in particular, requires bigger data size than other machine learning algorithms [14]. This paper uses available speech corpuses and attempts to design and develop automatic speech recognition for Amharic using neural networks that can fit into the existing speech corpuses and computing resources. The study also applies techniques such as stopping regularization [1, 15] to avoid model over-fitting, a

situation where the gap between training error and test error is too large [1].

The speech corpus used for this experiment has already been prepared and readily available by Solomon Teferra Abate *et al.* [2]. The speech corpus has training, validation and testing data sets and there is no need for dividing it into these data sets. The length of the speech corpus is 20 hours. The number of examples of the training, validation and test sets is 10,875, 400 and 400, respectively. Word error rate and character error rate are the standard evaluation techniques for ASR [16, 17]. Both of these techniques are used for this study.

2. Related Work

The first generation of studies in ASR for Amharic [6, 7, 18] used Hidden Markov Model (HHM) and HMM Took Kit called HTK for acoustic modeling.

Zegaye Seifu [19] developed large vocabulary, continuous speech and speaker independent Amharic speech recognizer using phone-based and tri-phone based recognizers. Speech corpuses consisting of 8000 sentences read by 80 people were recorded by Solomon Teferra Abate *et al.* [2]. A tri-phone based word recognition accuracy of 76.2% was achieved.

Solomon Teferra Abate and Menzel [6] used the language's distinct feature of the consonant-vowel (CV) syllable structure and the fact that its orthography has more or less a one to one correspondence with syllabic sounds. They developed a CV syllable-based speech recognizer using hidden Markov modeling and achieved 90.43% word recognition accuracy. Solomon Teferra Abate [8] designed HMM models using five emitting states and 12 Gaussian mixtures without skips and jumps and a word recognition accuracy of 90.43% was found.

Researchers using recognition for Amharic translation in [18] found encouraging results using morpheme-based language models and phoneme-based acoustic models with a recognition accuracy of 89.1%, 80.9%, 80.6%, and 49.3% at character, morph, word and sentence level, respectively.

Most of the attempts mentioned above for Amharic speech recognition were based on Hidden Markov Models (HMM). Two studies [7, 9] used a hybrid of HMM and neural networks with some improvement in terms of performance as compared to the HMM-based studies. However, the existing literature shows that recurrent neural networks are the central components of state-of-the-art speech recognition systems [10]. ASR researchers in other languages developed end-to-end speech recognition systems where all the components of the HMM paradigm are replaced with some form of neural network architecture with better performance [11, 12, 13]. Potential reasons for not designing and developing ASRs for Amharic that is based solely on neural network might be lack of enough speech data and computing resources. This study attempts to design and develop end-to-end speech recognition for Amharic using neural networks that can fit into the existing speech corpuses and computing resources.

3. The Proposed Solution

3.1 Design of the Solution

Audio data is transformed into spectrogram which is given to a convolutional neural network that is used for extracting features in the first stack of layers. The output of this layer is given as an input to layers of recurrent neural network. Taking inputs from the RNN layer, fully connected layers are employed to classify Amharic characters. Connectionist Temporal Classification (CTC) model is used to model encoding transcripts from the outputs of the fully connected layer. This stack of layers is presented in Figure 1. A raw audio for the string 'አ ወ ዳ ለ ሁ' [I like] is given at the bottom of Figure 1. The raw audio is first processed to get a spectrogram in the second layer. Once this input is passed through layers of recurrent neural network and an additional feedforward neural network, it will come up with a string (repeated characters and spaces included). Denoting output symbols of the softmax neurons by c, the probability of each symbol is given by $[\![p(c]\!]_i | O)$. An instance of a string of output from the given audio input is given in the upper-most layer. Because the probabilities of the different $c's$ are independent (i.e., $[\![p(c]\!]_2 = \text{አ} | O)$ and $[\![p(c]\!]_4 = \text{ወ} | O)$ are not

influenced to each other in any way). The probability of the whole string c will be:

P(c=ኣመዳለሁ || O)= p(c1=ኣ|O) p(c2=ኣ|O) p(c3= —|O) p(c4=መ|O) … p(c9=ሁ|O) p(c10=—|O)

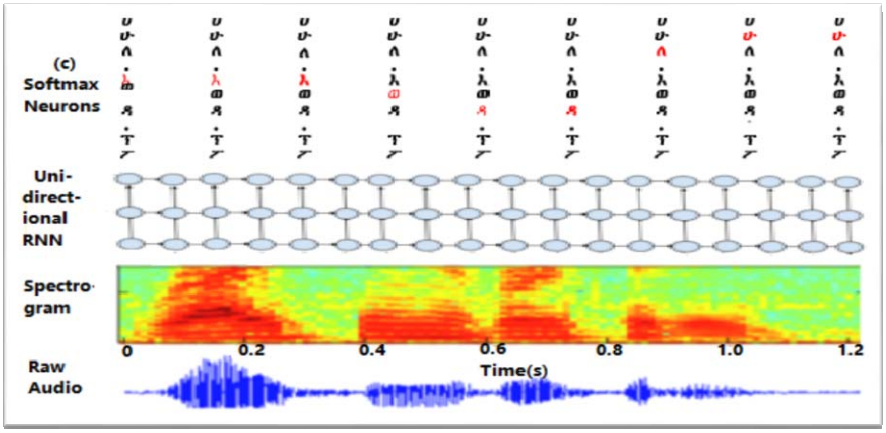

Figure 1: Speech Recognition System for Amharic

A mapping function such as $y \rightarrow \beta(c)$ can squize so that $\beta(ኣ\_መዳለሁ\_) = ኣመዳለሁ$. There are several instances of strings that have acoustic similarity. Out of these candidate strings, the CTC model picks the one that has the highest probability for transcription.

3.2 Implementation

As the audio for each of the examples is separately put in different audio files, a mapping should be made between these audio files and the corresponding texts. Duration of speech was also extracted from each audio file for purposes of efficiency so that input arrays are fine-tuned depending on available computing resources. A dictionary data structure was created to store audio directories, corresponding texts and duration of speech. This process was done to the training, validation and testing data sets.

To make Amharic characters, including blank spaces, manageable during CTC decoding, they were manually mapped to consecutive numbers starting from zero. The speed and accuracy of the connectionist temporal classification model depends on the computation and prediction of the probability of these characters. For this experiment, to save some computing resources, some rare complex characters that do not exist in the training, validation and testing sets were removed. Furthermore, only one character of each of the Amharic homophones is taken. For example the character "ሀ" [ha] is taken in place of "ሃ" "ሐ", "ሓ", "ኀ" and "ኃ". The character "ሰ" is taken in place of "ሠ", "ጸ" in place of "ፀ", and "አ" in place of "ኣ", "ኣ", "ዐ" and "ዓ".

3.3 The Experiment

a. Baseline Model Architecture

As a starting point for this experiment, a baseline model was specified following a model by previous research work in [12] for an end-to-end speech recognition in Manderin Chinese. In this baseline model, a spectrogram is given as an input. Following this spectrogram, a convolutional layer is laid, with 50 filters whose size is 11 each and a convolution stride of 2 and a valid convolution border mode. This is then followed by 3 layers of Gated Recurrent Units (GRUs), a variant of recurrent neural networks. Reflexive linear unit (ReLu) was used as an activation function. Number of epochs (i.e., the number of rounds the algorithm visits the whole data set) was fixed at 20. Finally, one fully connected layer is added to predict graphemes. A standard and reasonable choice of optimization algorithm is Stochastic Gradient Descent (SGD) with momentum with a decaying rate [1]. At the end of these layers of networks is the connectionist temporal classification loss function which predicts the string to compare with the ground truth. In the original baseline [12], the number of units (i.e., nodes) per layer is 1024. This is by far a very large network. However, given the huge size of data used and the computational power we have access to, this network size is not too large.

Given the above baseline model, adjustments were made on some of the hyper parameters based on the computing resources we have. These adjustments are related to the size of the network that has an impact on the computational burden of the training. The network size is reduced from 3 to 2. The number of units per layer is also adjusted to 100. While this number will increase for other variations of this baseline, this small network is designed to fit the small data sets we have. For quick iteration of training, the number of epochs was fixed to 20.

b. Baseline Results

Looking at the loss function of the baseline model in Figure 2, the average loss (over batches) decreases as the number of epochs increases for both training and validation data sets with varying degree of speed. This shows the model is learning and weight adjustments are being made.

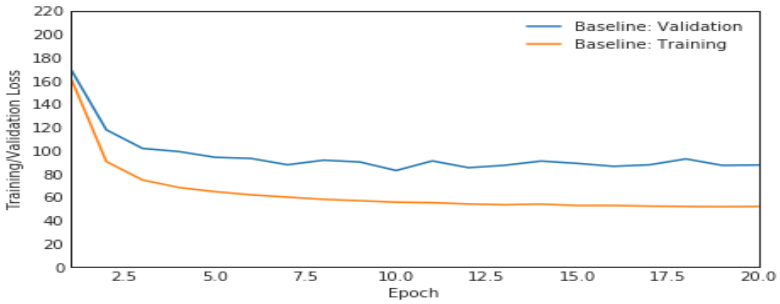

Figure 2: Loss Function for Baseline Model-Training and Validation

The fact that the model is still learning and both the training and validation losses tend to decline might indicate that increasing the number of epochs (i.e., training longer) will have a performance gain. The gap between validation and training loss diverges with the number of epochs. This shows the model is over fitting too quickly. That is, the model is using the training data too efficiently but training data does not generalize well to the validation data set. The next section will present a model with the number of epochs increased. Furthermore, as we will go through in detail later in the evaluation section, the divergence of the loss function between training and validation datasets is quantified as an indication of the high variance of the model which will be measured in terms of the difference in the performance of the model for the training and validation data sets.

3.2 Training Longer

There are various ways to design experiments that are based on the baseline by changing hyper parameters such as layers (number and type), initialization, learning rate, activation, number of epochs, batch normalization, batch size, etc. Table 1 presents these hyper parameters for the baseline model and changes in some of them generating different models.

Table 1: Changes in Hyper Parameters and Corresponding Models

Hyper parameter	*Baseline*	*Model-0*	*Model-1*	*Model-2*	*Model-3*
Convolutional layer					
No. of Conv. Layer	1	1	1	1	1
Filters	50	50	10	10	10
Kernel size	11	11	11	11	3
Conv stride	2	2	2	2	2
Conv border mode	valid	valid	valid	valid	valid
Recurrent layer					
Type of RNN	GRU	GRU	GRU	GRU	GRU
Number of RNN layers	2	2	2	3	3
Number units per layer	100	100	200	200	200
Optimization algorithms	SGD	SGD	SGD	SGD	SGD
Learning rate	0.02	0.02	0.02	0.02	0.02
Activation	RELU	RELU	RELU	RELU	RELU
Number of epochs	20	100	100	100	100

Hyper parameter	*Baseline*	*Model-0*	*Model-1*	*Model-2*	*Model-3*
Batch Normalization	Yes	Yes	Yes	Yes	Yes
Batch size	64	64	64	64	64

Model-0 is the same as the baseline except an increase in the number of epochs to 100. The rationale behind training longer lies in the further adjustments of weighs so that losses will decline and hence more performance gains. There is no systematic way for determining the number of epochs. Any arbitrary number can work as long as loss is declining and performance improved. For this variation (Model-0), number of epochs was fixed at 100. We can see from Figure 3 that while training loss has declined further, no significant loss reduction was observed for validation data set. It also seems the training loss quickly plateaus after the 17th epoch which might mean that we need to increase the size of the network [1].

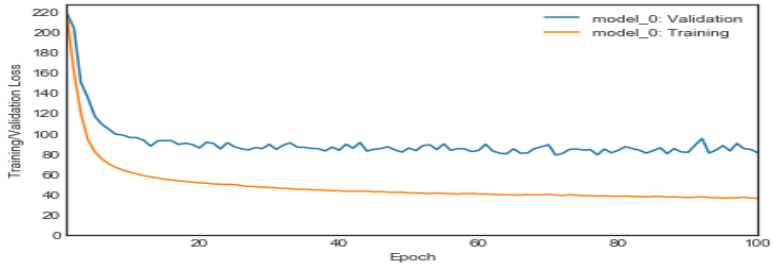

Figure 3: Loss Function for Baseline Model with More Number of Epochs

3.3 Changes in Network Size

An increase in the network size of neural networks can be thought of as an increase in the number of layers and number of hidden units (or nodes) per layer. Hence, two ways of increasing network size were made. Model-1 and Model-2 are presented in Table 1. Model-1 is similar in all aspects with the baseline model except the number of units per layer and number of epochs which have both increased. Model-2 is designed such that an additional layer of Gated Recurrent Units (GRUs) is added on Model-1. In both cases, the number of epochs increased to 100 iterations.

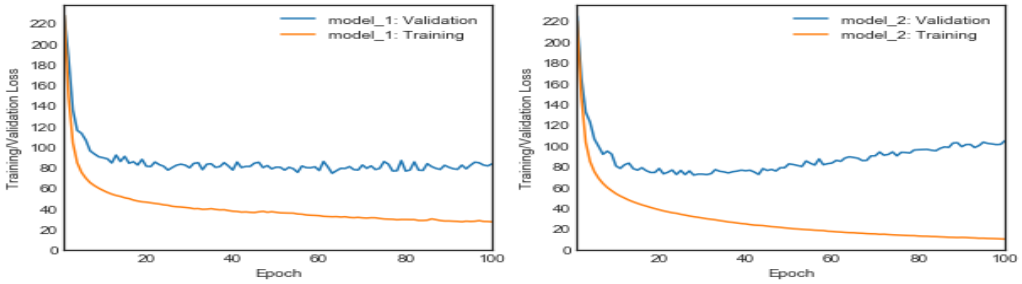

Figure 4: Loss Function for Baseline Model-Training and Validation

As shown in Figure 4, while more learning is happening in the training data in both models with little improvement for the validation data set, there is still more divergence between the training and validation loss. This divergence between training and validation loss increased for the model with a bigger network size (especially true for Model-2).

3.4 Early Stopping Regularization

Regularization is any modification made to a learning algorithm that is intended to reduce its generalization error but not its training error [1]. There are several forms of regularization techniques including dropout, L1, L2 and early stopping regularization. Dropout regularization, which looks at the units in a given layer and randomly drops some of the units using a given probability p of dropping out, has its own computational burden during training. The other regularization techniques, L1 and L2, should learn additional parameter called regularization parameter which adds to the computational burden of the training. Early stopping regularization, on the other hand, is the simplest and

least cost effective. It makes the training stop while there is no improvement in the decline of the validation loss function [15, 20]. Therefore, we used early stopping as a way out to reduce model over-fitting.

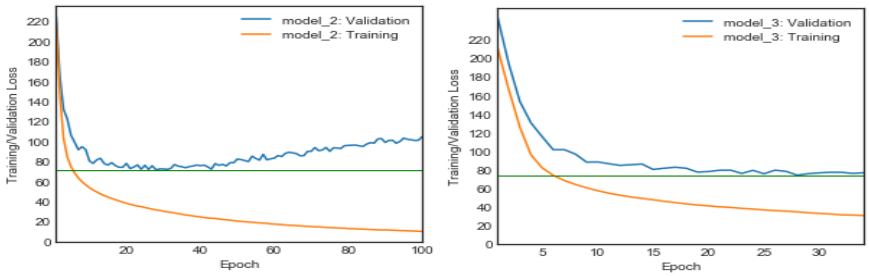

Figure 5: Early Stopping a Way out to Reduce Over-fitting

The first panel of Figure 5 repeats loss function of Model-2 explicitly showing the minimum of this loss function (indicated by the horizontal line). As can be seen at the 30th epoch loss function starts to increase diverging from the training loss. However, for the training with number of epochs fixed at 100, the final weights saved for the model are the ones at the 100th epoch. This is, however, worse than the weights at the 30th epoch. What early stopping does is to stop training (in this case at the 30th epoch depending on other parameters such as patience and the number of epochs with no improvement in validation loss) while loss fails to decline and save the model at that point.

4. Discussion

4.1 Recognizer evaluation

Table 2 presents the results for the five experiments including the baseline. These figures were computed as the average of the examples in the training, validation and testing datasets. Validation data was used to tune hyper-parameters while test data sets were used for final evaluation of the models.

Table 2: Performance Evaluation (Accuracy) Based on CER

Model	*Performance*		
	Training	*Validation*	*Test*
No. (share) of examples	10,875 (93%)	400 (3%)	400 (3%)
Baseline	62.5%	51.8%	52.5%
Model-0	75.0%	57.9%	59.1%
Model-1	79.7%	59.4%	59.2%
Model-2	92.9%	61.5%	62.1%
Model-3	75.8%	69.2%	69.6%

Performance for the baseline model was 62.5%, 51.8% and 52.5% for training, validation and test datasets, respectively. The best scenario is for Model-3 with 75.8%, 69.2% and 69.6% for training, validation and testing datasets, respectively. This model was trained with the maximum number of layers and with the computing resources we have access to. Furthermore, early stopping regularization was applied for this last model. The low level performance for training data set is an indication that training stops earlier than previous models. This early stopping, however, has an increasing impact in validation and testing data sets.

Performance based on Word Error Rate (WER) was computed corresponding to each of the models. WER for the five models is presented in Table 3. As expected, performance computed based on word error rate are lower than performance computed based on character error rate (i.e., word error rates are higher than character error rates for a given set of examples). The number of epochs from 20 to 100 has improved performance. Performance gain was also improved by increasing network size with 100 epochs. The best performing model based on WER is Model-3 where early stopping regularization is applied. Performance for training, validation, and testing data sets are 17.5%, 15.5% and 13.6%, respectively.

Table 3: Performance Evaluation (Accuracy) Based on WER

Model	*Performance*		
	Training	*Validation*	*Testing*
Baseline	9.7%	3.7%	4.1%
Model-0	24.5%	8.0%	9.1%
Model-1	32.1%	7.4%	7.4%
Model-2	57.6%	11.5%	11.7%
Model-3	17.5%	15.5%	13.6%
No. of words	78316	3591	3268

4.2 Results and Discussion

Previous studies [21] show that the move from HMM-based ASR systems to an ASR system based on neural networks has a performance gain. Furthermore, HMM-based models require features such as Mel-frequency ceptrum coefficients to be manually extracted and phonemes to be prepared. However, ASR systems with neural networks tend to work well with a lot of labeled data. Performance gain is achieved when we have large neural network or bigger data. Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with [1, 14].

In all of the models designed for the experiment, performance of training data set in terms of the declining loss function has improved. Furthermore, bias was shown to be declining with the number of epochs. This indicates the network is using the data efficiently. However, performance in cross-validation data set is poor. It improves across the different models but only with a small amount. In some instances (Model-2), it deteriorates after about 40 epochs. This is also testified by the performance of the cross-validation set. This divergence, called variance in the deep learning literature, is common in training of machine learning algorithms with limited size of data.

Typical recommendations for reducing variance include adding more training data (and cross-validation data), regularization such as dropout, L2, and early stopping regularization. Another recommendation is change in the architectures of the neural networks [1]. For this experiment, access to more data is not feasible at this stage. Regularization is a viable solution. An attempt to apply early stopping was made and better performance gain was achieved. Early stopping regularization was chosen because it does not add a significant computational burden as the other regularization techniques such as dropout, L1, and L2.

Collecting and preparing more speech data is the most viable solution. A study by Banko and Brill [14] found that algorithms (including perceptrons, earlier version of neural networks) in general work better with more data. The authors propose that, in the absence of enough speech corpuses, a logical step for the research community would be to direct efforts towards increasing the size of annotated training collections, while deemphasizing the focus on comparing different learning techniques trained only on small training corpora [14]. Increasing the complexity of neural networks, in most cases, requires having more data. This is mainly due to the number of parameters (i.e., weights) to be estimated during training. However, the rationale behind more data should be taken with caution. More data is promising for applications with inputs having enough features to be mapped with output and with enough number of parameters to be estimated. This implies collecting data that represents the real world and use more complex networks to capture the complexity of speech recognition.

5. Conclusion and Future Work

Using the available data, this study attempted to understand and investigate a possibility of designing and developing a read-speech, large vocabulary and speaker independent speech recognition system for Amharic with neural networks. For all the experiments that were conducted, the algorisms were able to learn both from the training and validation datasets as indicated by a declining loss function. On the other hand, variance increased with a declining bias and losses. With no hope of increasing size of speech data, increase in the number of epochs, a small increasing adjustment in the size of the network and early stopping regularization helped gain more performance.

Preparing more speech data is imperative for better performance gains. However, collecting a representative sample is crucial especially if the research output is applied to a real world application or if it is tested using a demo on an externally generated dataset. Data augmentation can be one way of increasing size of speech corpuses. Data augmentation is a process whereby new audio clips are generated from existing data sets. This way of increasing speech corpuses has an impact on the quality of the speech corpus to incorporate inputs that can be used for developing ASR in a noisy environment.

Constrained by computing resources, this study applied the default decoding algorithm. However, more robust search algorithms such as beam search and greedy search would be employed in the future for better performance. Furthermore, the results of this study were computed without using language model. Language modelling has been successfully employed in other researches on ASRs. Therefore, applying language models might help get more performance gains.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning", MIT Press. MIT, 2016.

[2] Solomon Teferra, W. Menzel, and B. Tafira, "An Amharic Speech Corpus for Large Vocabulary Continuous Speech Recognition," In Proceedings of the 9th Eur. Conf. Speech Commun. Technol., pp. 1601–1604, 2005.

[3] S. Ruan, J. O. Wobbrock, K. Liou, S. Corp, A. Ng, and J. A. Landay, "Comparing Speech and Keyboard Text Entry for Short Messages in Two Languages on Touchscreen Phones," Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol., Vol. 1, No. 4, pp. 1–23, 2017.

[4] Martha Yifru Tachbelie, Solomon Teferra Abate, and L. Besacier, "Using different acoustic, lexical and language modeling units for ASR of an under-resourced language - Amharic," Speech Commun., Vol. 56, No. 1, pp. 181–194, 2014.

[5] Michael Melese Woldeyohannis, L. Besacier, and Million Meshesha, "Amharic Speech Recognition for Speech Translation," 2016.

[6] Solomon Teferra Abate and W. Menzel, "Syllable-Based Speech Recognition for Amharic," in Semitic '07 Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, 2007, pp. 33–40.

[7] Hussein Seid and B. Gambäck, "A Speaker Independent Continuous Speech Recognizer for Amharic," Proc. 9th Eur. Conf. Speech Commun. Technol., pp. 3349–3352, 2005.

[8] Solomon Teferra Abate, "Automatic speech recognition for Amharic," Fchbereich Informatik der Universitat, Hamburg, 2005.

[9] Abebe Semie, "Developing a hybrid Hidden Markov Model/Artificial Neural Netowork based large vocabulary, speaker independent, continuous Amharic speech recognition," Bahir Dar University, 2015.

[10] A. L. Maas *et al*., "Building DNN acoustic models for large vocabulary speech recognition," Comput. Speech Lang., Vol. 41, pp. 195–213, 2017.

[11] W. Song and J. Cai, "End-to-End Deep Neural Network for Automatic Speech Recognition," CS224d Deep Learn. Nat. Lang. Process., pp. 1–8, 2015.

[12] D. Amodei *et al*., "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," Vol. 48, 2015.

[13] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," JMLR Workshop Conf. Proc., Vol. 32, no. 1, pp. 1764–1772, 2014.

[14] M. Banko and E. Brill, "Scaling to Very Very Large Corpora for Natural Language Disambiguation," 1998.

[15] A. Ng, "Deep learning specialization." Available at https://www.coursera.org /specialization/deep-learning, Last Accessed on 18, December 2018.

[16] D. Jurafsky and J. H. Martin, "Speech and Language Processing", 2nd Edition, Prentice Hall Series in Artificial Intelligence, 2009.

[17] P. Wang, R. Sun, H. Zhao, and K. Yu, "A New Word Language Model Evaluation Metric For Character Based Languages," in China National Conference on Chinese Computational Linguistics, 2013, pp. 1–10.

[18] Kinfe Tadesse, "Sub-word Based Amharic Word Recognition: an Experiment Using Hidden Markov Model (HMM)," 2003.

[19] Zegaye Seifu, "Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition," 2003.

[20] L. Prechelt, "Early Stopping-But When?".

[21] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs," Icassp 2011, pp. 4688–4691, 2011.