International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 6 Issue: 4 88 - 93

Recurrent Neural Networks for End-to-End Speech Recognition: A Comparative Analysis

Gauri Dhande
M.Tech.
The Department of Computer Engineering
K.J.Somaiya College of Engineering
Mumbai, Maharashtra 400077
Email:gauri.dhande@somaiya.edu

Prof. Zaheed Shaikh
The Department of Computer Engineering
K.J.Somaiya College of Engineering
Mumbai, Maharashtra 400077
Email: zaheedshaikh@somaiya.edu

***Abstract*—**Speech Recognition is correctly transcribing the spoken utterances by the machine. A new area that is emerging for the representation of the sequential data, such as Speech Recognition is Deep Learning. Deep Learning frameworks such as Recurrent Neural Networks(RNNs) were successful in replacing the traditional speech models such as Hidden Markov Model and Gaussian mixtures. These frameworks boosted the recognition performances to a large context. RNNs being used for sequence to sequence modeling, is a powerful tool for sequence labeling. End-to-End methods such as Connectionist Temporal Classification(CTC) is used with RNNs for Speech Recognition. This paper represents a comparative analysis of RNNs with End-to-End Speech Recognition. Models are trained with different RNN architectures such as Simple RNN cells(SRNN), Long Short Term Memory(LSTMs), Gated Recurrent Unit(GRUs) and even a bidirectional RNNs using all these is compared on Librispeech corpse.

Keywords - *Speech recognition, Recurrent Neural Networks, End-to-End model*

*****

I. INTRODUCTION

Speech is the basic form in which a human passes the information in voice form. Now-a-days, due to increase in use of electronic gadgets, recognizing the speech through machine has become an important aspect. Many physically challenged and visually impaired people can make use of such system. With the help of speech as an input, such people can use the technology and become expertise. However, speech recognition has to face challenges such as speech classes, speech styles, vocabulary, transducers, illness and channels; due to all this constraints the noise factor in automatic speech recognition is high[1].

Speech recognition involves analysis of speech signals to correctly identify the spoken words with the used of training its features. Speech signals are 1-D vector representation. Few of algorithms to extract the features from Artificial neural networks (ANN) are, linear prediction cepstrum coefficients (LPCC), Mel Frequency cepstrum coefficients (MFCC), combination of Linear Prediction coefficients and Mel Cepstrum coefficients (LPCMCC), the Support Vector Machine (SVM); combination of HMM and SVM etc [2]. One of the commonly used method to extract the feature from speech signal is Mel Frequency Cepstral Coefficient (MFCC). Being this a supervised system, text transcriptions along with speech signals are also passed in the form of input.

The flow chart of end-to-end system is shown in figure 1. This paper describes the comparison of RNN methods for end-to-end systems. The organization of this paper is as follows: Section I is the introduction for the topic. Section II is the literature survey for preprocessing method and Recurrent Neural Networks. Section III and Section IV is the introductory paragraphs for Connectionist Temporal Classification (CTC) and Decoding method used respectively. Section V is the comparatively analysis for different RNN architectures described in literature survey. Section VI is the future work followed by Section VII, which is conclusion for the paper.

II. LITERATURE SURVEY

Speech Recognition is transforming different speech signals into feature arrays, and these arrays are later supplied to the RNNs for training the model.

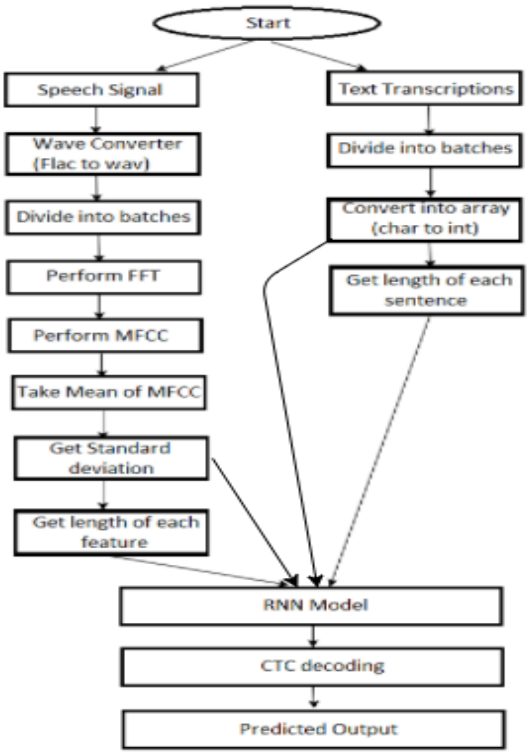


Figure 1. Algorithm

A. Method

1) *Data*

Librispeech corpse is English Speech corpse of 1000 hours derived from audio books that are part of the LibriVox project, and contains 1000 hours of speech sampled at 16kHz[3].

2) *Framing or Windowing*

Audio signals are non-stationary i.e. the properties changes very quickly. It is necessary to split them into frames where these are assumed to be stationary. This is known as windowing or framing. A window of 10ms is used in our project.

3) *Feature Extraction*

In this paper, the feature extraction technique call Mel Frequency Cepstral Coefficients (MFCC) is used. The general procedure of mel-cepstrum extraction actually involve, dividing the signal into frames, to obtain the power of spectrum, to convert the melspectrum and lastly uses the Discrete Cosines Transform (DCT) to get the cepstrum coefficient[4].

B. Recurrent Neural Networks (RNNs)

There are two kinds of neural networks, namely, feed-forward and feed-backwards. Feed-forwards are the networks where the data flows in forward direction i.e. from input to output and there are no backward connections. Feed-backwards are the networks where there are backward connections i.e. the current input data depends on the previous output.

RNNs are a type of feed-backward neural network. The output of the current timestamp depends on previous timestamp. There is memory assigned to every cell of RNN. This memory keeps the track of previously computed outputs. Thus, long term dependencies are taken care of.

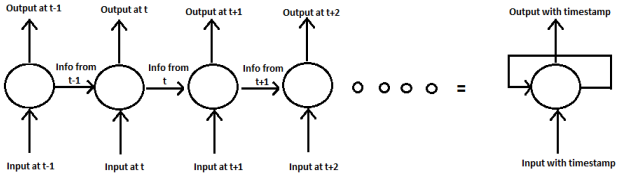


Figure 2. Recurrent Neural Networks

1) *Simple Recurrent Neural Network(SRNN)*

Simple RNN consists of three layer, input, output and hidden layer. All the information from the current timestamp is passed as current output and the input to the next timestamp.

Figure 3 shows the way weights are assigned to SRNN. There are two output equations, which are involved in SRNN.

First, the information from the previous timestamp(t-1) to next hidden unit in the next timestamp(t).

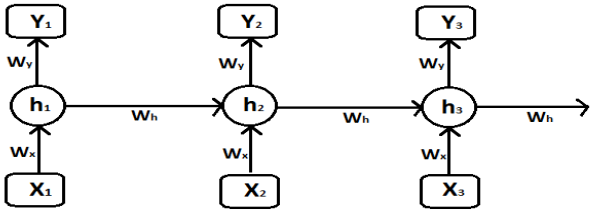


Figure 3. Simple Recurrent Neural Network

$$h_{(t)} = g_h(W_x X_{(t)} + W_h h_{(t-1)} + b_h)$$

Thus, current hidden layer is a summation of current input multiplied by its weights, plus info from the hidden layer of previous timestamp plus biases, applied by the activation function. Second, the output at that particular timestamp

$$Y_{(t)} = g_y(W_y h_{(t)} + b_h)$$

Thus, it is the summation of current hidden layer multiplied by its weights, plus biases, applied by the activation function. Where, W and b are weights and biases respectively, and g is the activation function.

Although Simple RNNs are capable of learning sequential time series patterns, they have vanishing and exploding gradient problem that affects their performance. Also, they can't carry out long term dependencies of sequential data.

2) *Long Short Term Memory(LSTM)*

The problem of vanishing and exploding gradient is solved by LSTMs. The idea behind is the short term memory, which has to last for a long time. LSTM is the abstraction of computer memory that works along with recurrent neural networks. LSTMs have special blocks, known as gates, allows only a specific amount of information to be gated in and then the information to be gated out. Within intermediate time, these gates are closed, so that other information doesn't interfere.

Input gate: Responsible for how much of the output data to be written in memory cell. It receives the input data as well as the information from the previous cell.

Output gate: Responsible for sending information from LSTM back to recurrent network. It receives the input data and the data from the memory cell.

Forget gate: Responsible for the data maintenance in memory cell. It receives the same input data from the gates and the networks, but calculates how much data to remember.

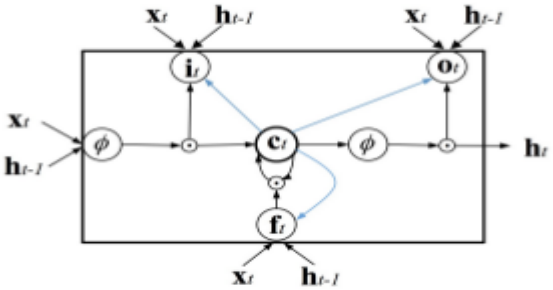


Figure 4. A memory block of LSTM[5]

The equations involved for the timestamp t are:

$$i_{(t)} = g_h(W_{ix}X_{(t)} + W_{it}h_{(t-1)} + W_{ic}c_{(t-i)} + b_i)$$
$$f_{(t)} = g_h(W_{fx}X_{(t)} + W_{fh}h_{(t-1)} + W_{fc}c_{(t-1)} + b_f)$$
$$c_{(t)} = f_{(t)}oc_{(t-1)} + i_{(t)}\Theta \text{ ᔕ}(W_{cx}X_{(t)} + W_{ch}h_{(t-1)} + b_c)$$
$$o_{(t)} = g_h(W_{ox}X_{(t)} + W_{oh}h_{(t-1)} + W_{oc}c_{(t)} + b_o)$$
$$h_{(t)} = o_{(t)}\Theta \text{ ᔕ}(c_{(t)})$$

where $i_{(t)}$, $o_{(t)}$, $f_{(t)}$, and $c_{(t)}$ are the outputs of the input gate, output gate, forget gate and memory cells respectively. The W_x weight matrices connect the inputs with the units, whereas the W_h weight matrices connect the previous memory cell states with the units, the W_c terms are diagonal weight matrices for peephole connections. Also, g_h is the logistic sigmoid non-linearity, and ᔕ is the hyperbolic tangent nonlinearity[5].

3) *Gated Recurrent Unit(GRU)*

Gated Recurrent Unit (GRUs) is a variation of LSTM. The difference between LSTMs and GRUs, is LSTM have four gates to control its memory contains whereas GRU have two gates to control its memory contains. As a result, GRU is easier to train.

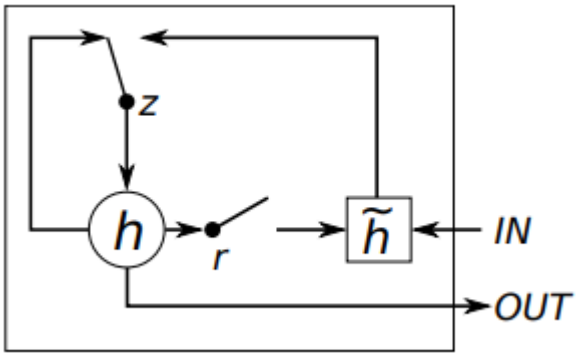


Figure 5. Gated Recurrent Unit[6]

GRU have two gates and does not have the memory cell[7].

Update gate: Responsible for holding the information of previous timestamp. It receives the input of the current timestamp.

Reset gate: Responsible for the amount of information that is to forget.

The equations involved for the timestamp t are:

$$z_{(t)} = g_h(W_zX_{(t)} + U_zh_{(t-1)})$$
$$r_{(t)} = g_h(W_rX_{(t)} + U_rh_{(t-1)})$$
$$\hat{h}_{(t)} = \tanh(W_xX_{(t)} + r_{(t)}\ \Theta\ Uh_{(t-1)})$$
$$h_{(t)} = z_{(t)}\ \Theta\ h_{(t-1)} + (1 - z_{(t)}\ \Theta\ h_{(t)})$$

Where, $z_{(t)}$ and $r_{(t)}$ are the outputs of update gate and read gate respectively. $W_{(z)}$ and $W_{(r)}$ are the weight matrices in reference to update gate and read gate respectively. $\hat{h}$ a memory content which will use the reset gate to store the previous information, $h_{(t)}$ is vector that holds current information and passes it down to the network.

4) *Bidirectional Recurrent Neural Network(BRNN)*

One of the limitations of traditional RNN is that they are only able to use previous timestamp information. In bidirectional RNN, the input information can be trained from both past and future timestamp. The states of the RNN are split into positive direction i.e forward direction and negative direction i.e backward direction. Finally, positive direction and negative direction outputs are added and provided to the next layer.

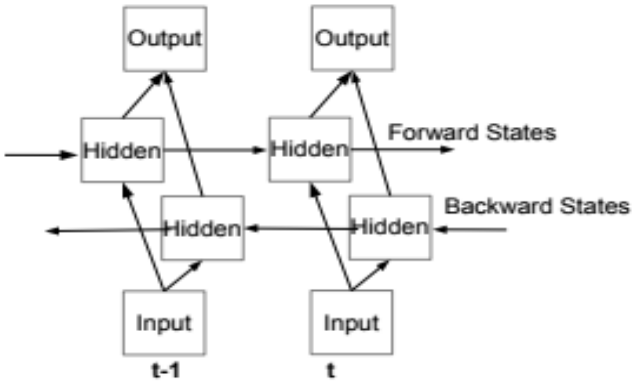


Figure 6. Bidirectional Recurrent Neural Network[8]

The equations involved for the timestamp t are:

$$h^{F}_{(t)} = g_h(W^{F}_{xh}X_{(t)} + W^{F}_{hh}h^{F}_{(t-1)} + b^{F}_{h})$$
$$h^{B}_{(t)} = g_h(W^{B}_{xh}X_{(t)} + W^{B}_{hh}h^{B}_{(t-1)} + b^{B}_{h})$$
$$Y_{(t)} = W^{F}_{hy}h^{F}_{(t)} + W^{B}_{hy}h^{B}_{(t)} + b_y$$

Where, $h^{F}_{(t)}$ is the forward hidden output, $h^{B}_{(t)}$ is the backward hidden output and $Y_{(t)}$ is the final output of the bidirectional recurrent layer.

III. CONNECTIONIST TEMPORAL CLASSIFICATION (CTC)

Speech recognizer converts the sequences of utterances into a sequences of words. Input sound utterances may be much longer than output words, i.e. the input and output will be of different lengths, which creates a problem for our standard RNN architecture. Thus, Connectionist Temporal Classification (CTC) comes to rescue.

There is always a blank symbol associated with CTC. So, for k number of graphemes, we have k+1 total classes. With all these decision, the network will decide whether to output the alphabet or a blank symbol.

Connectionist Temporal Classification (CTC) works with Softmax function. Softmax layer defines the probability output distribution Pr(k|t) at every timestamp along with input [9].

IV. DECODING

Before decoding the network refers to the process by which the model finds the most probable output transcription y given an input x. Here, CTC decoding method is used. The rule used for decoding is: Given a specific character sequence c, squeeze out duplicates and blanks to yield transcriptions [10].

V. COMPARATIVE ANALYSIS

This section consists of two parts: First the part is training and second is analysis.

A) Training

The RNN Models is trained on GPU GeForce GTX 1060 6GB. RNN is trained on 'train-clean-100' of 100 hours set and validated on 'dev-clean'. All the Models consists of common hyper-parameters such as learning rate of 0.001 and momentum with 0.99. The optimizer used is Stochastic Gradient Descent(SGD) with nesterove value true, delay with value (1e-6) and clipnorm with value 5. The network of RNN layers consists of 3 and there are 200 cell units for each layer. Number of epochs are 20 with mini-batch size of 10. There are total of 28 characters that has to be recognized in the dataset (26 alphabets, apostrophe ('), space). Batch normalization and dropout techniques are used to avoid overfitting of the model.

B) Analysis and Graphs

The metrics for evaluation is loss and mean square error(mse). The model with the numbers is described below:
model_0: A Simple RNN model.
model_1: A Long Short Term Memory model.
model_2: A Gated Recurrent Unit model.
model_3: A Bidirectional Simple RNN model.
model_4: A Bidirectional Long Short Term Memory model.
model_5: A Bidirectional Gated Recurrent Unit model.

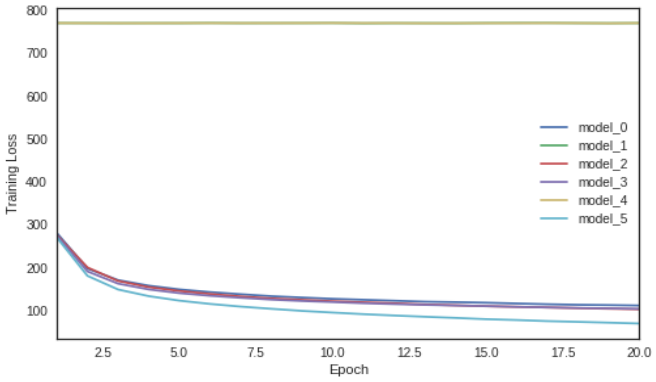


Figure 7. Epoch vs Training Loss Graph

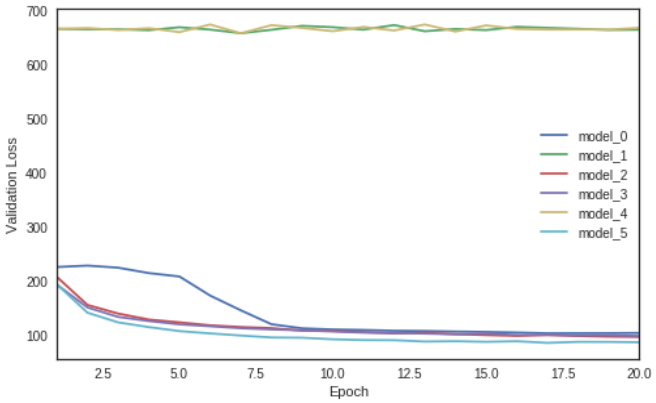


Figure 8. Epoch vs Validation Loss Graph

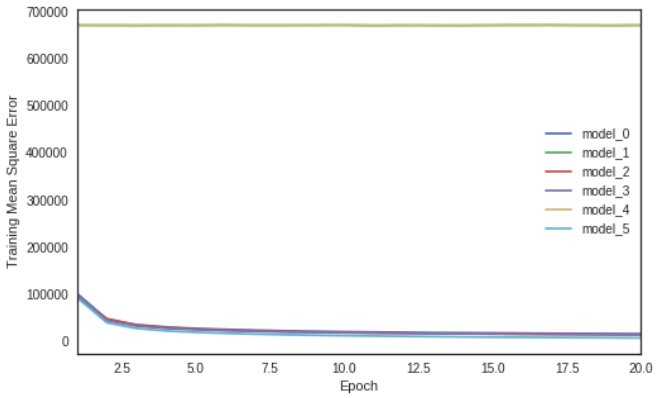
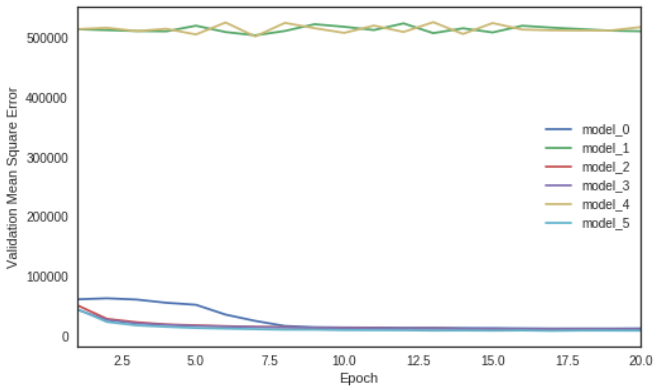


Figure 9. Epoch vs Training Mean Square Error Graph



Figue 10. Epoch vs Validation Mean Square Error Graph

Model_0 and model_3 have almost same loses. Simple RNN and bidirectional simple RNN does not show any changes with respect to all the metrics. Increasing the number of epoch makes a significant difference for both models

Model_1 and model_4 (both variant of LSTMs) have constant loss, can be improved by changing model capacity and regularization techniques.

Model_2 and model_5, a GRU implementation, shows a large difference between the losses. Model_5 gave the best results for among all the models, in every metrics. These results can be further improved by increasing the number of epochs.

	trained loss	trained mse	validation loss	validation mse
model_0	111.646	14990.1	104.336	13686.1
model_1	770.269	671984	664.867	510042
model_2	102.964	12855.1	97.0203	11871.8
model_3	103.952	13129.5	98.8551	12433.4
model_4	770.382	672207	668.245	517052
model_5	69.7156	6100.63	87.0962	9984.63

Table1: A table of comparison for all models and their metrics.

VI. CONCLUSION AND FUTURE WORK

This paper demonstrates various RNN architecture on a common hyper-parameters. Training loss can be further reduced by increasing the number of epochs. The validation loss can be reduced with hyper-parameters tuning. To make the results easy to produce and to compare, all networks are implemented on the common Keras, networks API, written in Python with TensorFlow as backend. A bidirectional version of GRU gave the best results with least training loss of 69.71. Future work will comprise of effective training of RNNs with more efficient methods, for example, the use of convolution neural networks with RNNs[13]. In addition, using of more comprehensive features [11] and using a different RNN training toolkit [12] inducing more advanced features.

REFERENCES

[1] Chavan, K. and Gawande, U., 2015, February. Speech recognition in noisy environment, issues and challenges: A review. In *Soft-Computing and Networks Security (ICSNS), 2015 International Conference on* (pp. 1-5). IEEE.

[2] El Ayadi, M., Kamel, M.S. and Karray, F., 2011. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*, *44*(3), pp.572-587.

[3] On, C.K., Pandiyan, P.M., Yaacob, S. and Saudi, A., 2006, June. Mel-frequency cepstral coefficient analysis in speech recognition. In *Computing & Informatics, 2006. ICOCI'06. International Conference on* (pp. 1-5). IEEE.

[4] Miao, Y., Gowayyed, M. and Metze, F., 2015, December. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on* (pp. 167-174). IEEE.

[5] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[6] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[7] Irie, K., Tuske, Z., Alkhouli, T., Schluter, R. and Ney, H., 2016. *LSTM, GRU, highway and a bit of attention: an empirical overview for language modeling in speech recognition*. RWTH Aachen University Aachen Germany.

[8] Arisoy, E., Sethy, A., Ramabhadran, B. and Chen, S., 2015, April. Bidirectional recurrent neural network language models for automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (pp. 5421-5425). IEEE.

[9] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 6645-6649). IEEE.

[10] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J., 2006, June. 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.

[11] Yao, K., Zweig, G., Hwang, M.Y., Shi, Y. and Yu, D., 2013, August. Recurrent neural networks for language understanding. In Interspeech (pp. 2524-2528).

[12] Mikolov, T., Kombrink, S., Deoras, A., Burget, L. and Cernocky, J., 2011, December. Rnnlm-recurrent neural network language modeling toolkit. In Proc. of the 2011 ASRU Workshop (pp. 196-201).

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