Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU

Apeksha Shewalkar, Deepika Nyavanandi, Simone A. Ludwig
Department of Computer Science, North Dakota State University,
Fargo, ND, USA

January 23, 2019

Abstract

Deep Neural Networks (DNN) are nothing but neural networks with many hidden layers. DNNs are becoming popular in automatic speech recognition tasks which combines a good acoustic with a language model. Standard feedforward neural networks cannot handle speech data well since they do not have a way to feed information from a later layer back to an earlier layer. Thus, Recurrent Neural Networks (RNNs) have been introduced to take temporal dependencies into account. However, the shortcoming of RNNs is that long-term dependencies due to the vanishing/exploding gradient problem cannot be handled. Therefore, Long Short-Term Memory (LSTM) networks were introduced, which are a special case of RNNs, that takes long-term dependencies in speech in addition to short-term dependencies into account. Similarily, GRU (Gated Recurrent Unit) networks are an improvement of LSTM networks also taking long-term dependencies into consideration. Thus, in this paper, we evaluate RNN, LSTM and GRU in order to compare their performances on a reduced TED-LIUM speech data set. The results show that LSTM achieves the best word error rates, however, the GRU optimization is faster while achieving word error rates close to LSTM.

1 Introduction

Deep learning is a term used to describe a specific class of artificial neural networks one that is composed of many layers. Neural networks for short, have existed for many decades. However, the training of deep architectures had failed until Geoffrey Hinton's breakthrough work in 2006 [1]. Even though there are some algorithmic innovations to these deeper networks, however, the dramatic rise in computing power using GPUs made the processing of larger data sets possible.

Deep learning methods have many application areas, and many successes have been seen in particular in the image processing area. For example, a deep learning architecture called Convolutional Neural Networks (CNNs) are designed to emulate the behavior of a visual cortex. CNNs perform very well on any visual recognition tasks. The CNN architecture consists of special layers called convolutional layers and pooling layers. These layers allow the network to encode certain images properties.

Autoencoders is another class of a deep learning architecture. Autoencoders are used to reduce the input data by decreasing the dimensionality of the feature space. In stacked denoising autoencoders for example, a partially corrupted output is cleaned, i.e., de-noised.

Another area where deep learning is successfully applied is automatic speech recognition. In automatic speech recognition, good acoustic and language models are combined [2, 3]. The speech recognition problem involves time series data.

Feedforward neural networks are unidirectional whereby the outputs of one layer are forwarded to the following layer. These feedforward networks cannot persist past information. Furthermore, when DNNs are used to analyze speech recognition, certain issues are encountered including different speaking rates, and temporal dependencies [4, 5, 6]. DNNs can only model fixed size sliding window of acoustic frames but cannot handle different speaking rates [6]. Recurrent Neural Network (RNN) is another class of network that contains loops in the hidden layer to retain the information at the previous time step to predict the value of the current time step. This mechanism helps RNNs to handle different speaking rates [6].

Temporal dependencies could be an issue while analyzing speech recognition tasks. Temporal dependencies may be present in the short-term or long-term depending on the speech recognition problem. RNNs take into account only the short-term dependencies due to the vanishing/exploding gradient problem. In the last couple of years, RNNs are being applied to a variety of problems such as machine translation, image captioning, and speech recognition. Speech involves a dynamic process, and thus, RNN seems a good choice over traditional feedforward network [7].

However, the applicability of RNN is limited due to two reasons. The first is that RNNs require pre-segmented training and post-processing of the output to convert it into labeled sequences. The Connectionist Temporal Classification (CTC) for labeling sequence data in training with RNN solves this shortcoming. The CTC method has been proven to be helpful where alignment between input and output labels is unknown [8]. Second, for long term dependencies in data where the gap between the relevant information and the place where it is needed is large, RNNs have only limited use. Thus, a special type of RNN, Long Short-Term Memory (LSTM) networks have been introduced. LSTMs are designed to work on long-term dependencies in data. LSTM prove to be effective in speech recognition tasks [7] where the special memory cells of LSTMs are used to identify long dependencies. Slightly different than LSTM is the Gated Recurrent Unit (GRU) introduced in 2014 [9]. GRUs are also designed for long-term dependencies and work well with sequential data as do LSTMs.

In this paper, we have evaluated the performance of RNN, LSTM, and GRU for speech recognition applied to the reduced TED-LIUM data set [10] using an appropriate regularization method. Different parameterized models are trained end-to-end using CTC for labeling sequence data.

The paper is structured as follows. Section 2 describes related work in the area of speech recognition and deep learning. In Section 3 the three approaches applied are described. Section 4 describes the experiments conducted and discusses the results. The conclusion and future work is provided in Section 5.

2 Related Work

Traditionally, generative models were used for speech recognition. Generative models are typically composed of Maximum-A-Posteriori (MAP) estimation, Gaussian Mixture Models (GMMs), and Hidden Markov Models (HMM) [11, 12]. These traditional models require expert knowledge (i.e., knowledge of a specific language) as well as preprocessing of the text for Automatic Speech Recognition (ASR) [12]. Fortunately, an end-to-end ASR does not require expert knowledge because it depends on paired acoustics and language data [12].

Recent advances in deep learning have given rise to the use of sequence-to-sequence models (discriminative models) for speech recognition [11, 13]. In simple terms, sequence-to-sequence model for speech recognition takes an acoustic sequence as input and returns a transcript sequence as output [11].

Our work is guided by many previous works done in this area. In particular, RNNs have achieved excellent results in language modeling tasks as outlined in [14, 15]. Language modeling is based on a probabilistic model that is fitted to assign probabilities to sentences. This is accomplished by predicting the next word in a sentence given previous word data. The experiments were performed on the popular Penn Tree Bank (PTB) data set [15].

Language modeling is key to many problems such as speech recognition, machine translation or image captioning. RNNs and LSTMs have both been used to map sequences to sequences as well. Sequence to sequence models are basically made up of two RNNs; an encoder to process the input and a decoder to generate the output. In [16], multi-layered RNN cells have been used for the translation tasks and were evaluated on a popular English to French translation from the WMT'14 data set [16].

LSTM network architectures have proven to be better than standard RNNs on learning Context Free Language (CFL) and Context Sensitive Language (CSL) as described in [17]. Particularly, in speech recognition tasks, RNNs and LSTMs have achieved excellent results. Sequence labeling is an important task in training RNNs during the speech recognition process. HMM-RNN frameworks were used in the past [18, 19], however, they do not perform well with DNN.

Graves et al. [8] came up with a very efficient Connectionist Temporal Classification (CTC) method of sequence labeling to train RNNs end-to-end. This method works very well for problems where input-output label alignment is unknown, and the method does not require presegmented training data and post processing of the outputs. In addition, Graves et al. also introduced deep LSTM RNNs and evaluated the framework on speech recognition. Particularly, RNN with CTC was used to train the model end-to-end. The authors achieved the best recorded score on the TIMIT phoneme recognition benchmark [7].

Sak et al. [20] evaluated and compared the performance of LSTM, RNN and DNN architectures on a large vocabulary speech recognition problem - the Google English Voice Search Task. The authors have used an updated architecture compared to standard LSTM to make better use of the model parameters.

Several experiments are performed on the TIMIT speech data set using bidirectional LSTM, deep bidirectional LSTMs, RNNs, and hybrid architectures. Bidirectional LSTM networks have been used for the phoneme classification task [21, 22]. Results have shown that bidirectional LSTM performs better than unidirectional LSTM and standard RNNs on the frame-wise phoneme classification task. These results suggest that bidirectional LSTM is an effective architecture for speech processing where context information is very important.

A hybrid bidirectional LSTM-HMM system applied to phoneme recognition has proved as an improvement over unidirectional LSTM-HMM as well as traditional HMM systems. Bidirectional LSTMs have been experimented on the handwriting recognition problem, and were evaluated on both online and offline data, thus proving to outperform the state-of-the-art HMM based system [23].

Deep bidirectional LSTMs have been applied to the speech recognition task whereby each hidden layer was replaced by a combination of a forward layer and a backward layer. In this architecture, every hidden layer receives input from both the forward and backward layer at one level below. This type of deep bidirectional LSTM network combined with the CTC objective function has been used for end-to-end speech recognition. The network was evaluated on the Wall Street Journal corpus [24]. The authors' novel approach with the objective function has allowed direct optimization of the word error rate, even in the absence of a language model. A hybrid of deep bidirectional LSTM and HMM system has been used for speech recognition on the TIMIT data [25] outperforming GMM deep network benchmarks on part of the Wall Street Journal corpus.

Different DNN architectures have been evaluated on hundreds of hours of speech data in recent years like Wall Street Journal, Librispeech, Switchboard, TED-LIUM, Fisher Corpus [26, 27, 28]. Specifically, the TED-LIUM data set has been used in a variety of tasks such as in audio augmentation [28], in modeling probabilities of pronunciation and silence [29], etc. Furthermore, the TED-LIUM data set has also been used in automatic speech recognition combined with human correction at the word level and lattice level [30].

Gated Recurrent Units (GRU) recurrent neural networks were introduced by Cho et al. in 2014 [31]. GRUs are similar to LSTMs, both were designed to handle long term dependencies. However, GRUs have a simpler structure than LSTMs. Both architectures have been used for polyphonic music modeling as well as for speech recognition tasks [9, 32]. The results show that GRUs are equally efficient as LSTMs.

Our work is similar to Baidu research [26] where a bidirectional RNN is used to speed up the speech recognition performance using GPU (Graphics Processing Unit) parallelization. We have used a single GPU structure for our experiments to evaluate and compare the results of three recurrent networks, namely bidirectional RNN, bidirectional LSTM, and bidirectional GRU.

3 Recurrent Neural Network Architecture

This section first outlines the three models used for the experimentation followed by the chosen architecture description.

3.1 Recurrent Neural Network (RNN)

As mentioned earlier, in RNN the decision made at time $t-1$ affects the decision at time t. Thus, the decision of how the network will respond to new data is dependent on two things, (1) the current input, and (2) the output from the recent past. RNN calculates its output by iteratively calculating the following two equations:

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

where, x is the inputs, y is the output sequence, h is the hidden vector sequence at time slices $t = 1$ to T. Further W represents the weight matrices, and b represents the biases. $\mathcal{H}$ is the activation function used at the hidden layers.

3.2 Long Short-Term Memory (LSTM)

As a solution to the shortcomings of normal RNNs, Hochreiter and Schmidhuber came up with LSTM networks [33]. Special memory cell architecture in LSTM make it easier to store information for long period. The cell structure has been modified by many people since then, however, the standard formulation of a single LSTM cell can be given by following equations:

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

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * tanh(C_t) \quad (8)$$

where σ is the sigmoid function, $tanh$ is the hyperbolic tangent function, i, f, o, C, $\tilde{C}$ are the input gate, forget gate, output gate, memory cell content, and new memory cell content, respectively. The sigmoid function is used to form three gates in the memory cell, whereas the tanh function is used to scale up the output of a particular memory cell.

3.3 Gated Recurrent Units (GRU)

Introduced in 2014 [9], GRUs are similar to LSTMs but they have fewer parameters. They also have gated units like LSTMs which controls the flow of information inside the unit but without having separate memory cells. Unlike LSTM, GRU do not have output gate, thus exposing its full content. GRU formulation can be given by following equations:

$$r_t = sigm(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad (9)$$

$$z_t = sigm(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \quad (10)$$

$$\tilde{h_t} = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad (11)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t} \quad (12)$$

where r_t, z_t, x_t, h_t are the reset gate, update gate, input vector and output vector, respectively. Similar to LSTM, W denotes the weight matrices, b the biases, $sigm$ is the sigmoid activation, and $tanh$ is the hyperbolic tangent activation function.

Both LSTM and GRU are equally capable to handle long term dependencies, and have been experimented and compared with on machine translation tasks and proved to be comparably efficient [34].

3.4 Architecture for Experiments

The RNN architecture that we are using is based on [26]. The architecture takes speech spectrograms as an input and creates English text as an output. During pre-processing, a typically small window of a raw audio waveform is taken to compute FFT (Fast Fourier Transform) to calculate the magnitude (power) in order to describe the frequency content in the selected local window. Then, the spectrogram is generated by concatenating frames from adjacent windows of the input audio. These spectrograms serve as the input features for the RNN.

Consider a single input utterance x and label y as being sampled from the training set $X = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ...\}$ where every utterance $x^{(i)}$ is a time series of length $T^{(i)}$; $T^{(i)}$ is a time-slice represented as a vector of speech features $x_t^{(i)}$ from $t = 1, ..., T^{(i)}$. Eventually, the sequence of inputs x are converted into the sequence of character probabilities for transcription y as: $\hat{y}_t = \mathbb{P}(c_t|x)$, where $c_t \in \{a, b, c, ..., z, space, apostrophe, blank\}$.

The RNN model consists of one input layer, one output layer and five hidden layers. Figure 1 shows the architecture with the different layers and notation used below. The hidden layer is denoted by $h^{(l)}$. Thus, for input x, $h^{(0)}$ is the input and the output at the input layer depending on the spectrogram frame x_t and context C of frames where t is the time. The first three of the five hidden layers are normal feedforward layers. For each time t, these are calculated by:

$$h_t^{(l)} = g(W^{(l)} h_t^{(l-1)} + b^{(l)}) \tag{13}$$

In Equation 13, $g(z)$ is a clipped rectified linear unit (ReLu) activation function that is used to calculate the output at hidden layers, $W^{(l)}$ represents the weight matrix, and $b^{(l)}$ is the bias vector at layer l. In order to avoid the vanishing gradient problem, ReLu functions are chosen over sigmoid functions.

The following layer (layer 4) is a Bidirectional Recurrent layer (BRNN) consisting of one forward hidden sequence and one backward hidden sequence [7]. Standard RNNs make use of only the previous context information but bidirectional RNN explore the future context as well. In particular, for speech where complete utterances are recorded at once, BRNN is a better choice over simple RNN. The set of two hidden sequence layers in BRNN, one forward recurrent sequence $h^{(f)}$, and one backward hidden sequence $h^{(b)}$ can be formulated as:

$$h_t^{(f)} = g(W^{(4)} h_t^{(3)} + W_r^{(f)} h_{t-1}^{(f)} + b^{(4)}) \tag{14}$$

$$h_t^{(b)} = g(W^{(4)} h_t^{(3)} + W_r^{(b)} h_{t+1}^{(b)} + b^{(4)}) \tag{15}$$

Please note that the forward sequence is calculated from $t = 1, ..., t = T^{(i)}$ for the i^{th} utterance, whereas the backward sequence is calculated from $t = T^{(i)}, ..., t = 1$. This way BRNN processes the data in both directions using two separate hidden sequences, and then 'feedforward' the output to the next layer, which is layer 5. Thus,

$$h_t^{(5)} = g(W^{(5)} h_t^{(4)} + b^{(5)}) \tag{16}$$

where $h_t^{(4)} = h_t^{(f)} + h_t^{(b)}$.

Our experiments are performed with three models, one where we use bidirectional RNN, the second with bidirectional LSTM, and the last model with a bidirectional GRU layer. The formulation of bidirectional LSTM or GRU is the same as bidirectional RNN, however, instead of RNN we use either a LSTM cell or a GRU cell in layer 4.

For the output layer a standard softmax function is used in order to calculate the predicted probabilities of characters for each time slice t, and character k in the alphabet. This output is given by:

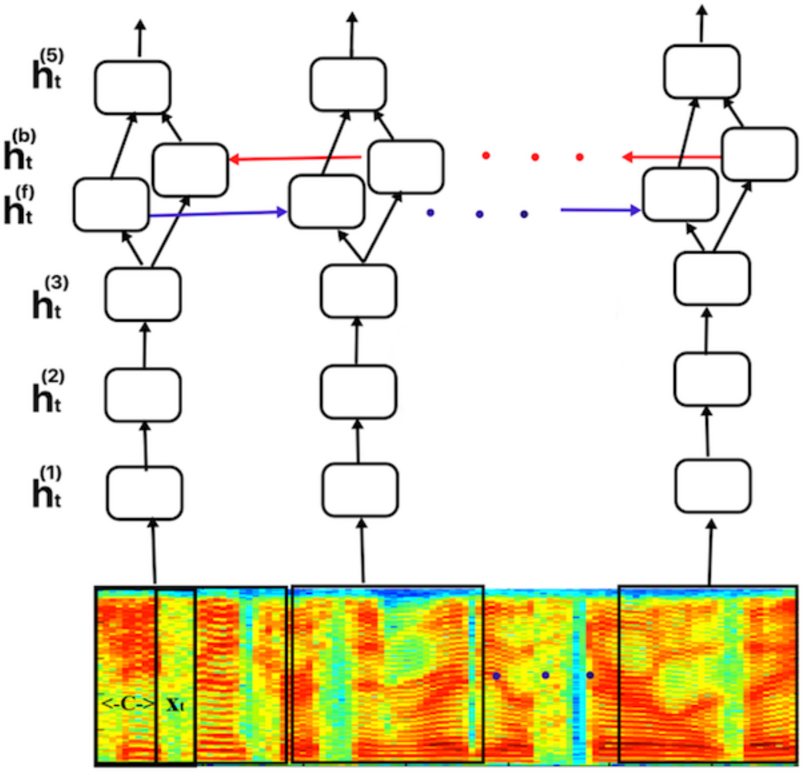

Figure 1: RNN Architecture

$$h_{t,k}^{(6)} = \hat{y}_{t,k} \equiv \mathbb{P}(c_t = k|x) = \frac{exp(W_k^{(6)} h_t^{(5)} + b_k^{(6)})}{\sum_j exp(W_j^{(6)} h_t^{(5)} + b_j^{(6)})} \tag{17}$$

where $W_k^{(6)}$ and $b_k^{(6)}$ denote the k^{th} column of the weight matrix and the k^{th} bias, respectively.

After the character probabilities are calculated, the CTC loss is calculated next [26]. The CTC loss function is used to integrate over all possible alignments of characters. Thus, given the output of the network, the CTC loss function calculates the error of the predicted output as the negative log likelihood of probability of the target. For this, the network output from Equation 17, the probabilities of the alphabet over each time frame, is the input to the CTC loss function. Given the definition of the CTC loss, the gradient of the loss according to its inputs have to be calculated. This loss can then be 'backpropagated' to the weights of the network. The ADAM optimization algorithm [35] has been used for the backpropagation training since this training algorithm is very tolerant to learning rate as well as to other training parameters, and thus, requires less fine-tuning.

4 Experiments and Results

In this section, first the data set is described as well as the evaluation measures used for the experiments are listed, followed by the results and discussion.

4.1 Data Set

We have used a subset of the improved TED-LIUM release 2 corpus [10]. The latest version of the TED-LIUM data set has an improved language model which is an important factor in achieving reduced WER (Word Error Rate) values [2]. The data set is publicly available and contains filtered data having audio talks and their transcriptions obtained from the TED website. This corpus is particularly designed to train acoustic models. For our experiments, we have reduced the data set of size 34.3 GB to 11.7 GB. The data set is available at [36]. The data set has separate data folders for training, validation and testing. The following is contained in the data set:

- 378 audio talks in NIST sphere format (SPH)
- 378 transcripts in STM format
- Dictionary having pronunciation (152k entries)
- Improved language model having selected monolingual data from WMT12 corpus [2]

4.2 Evaluation Measures

In speech recognition, there are two different types of performance or evaluation measures, which are based on (1) accuracy, and (2) speed [37]. Evaluation measures based on accuracy include WER, loss, and mean edit distance.

WER is the most commonly used error measurement in ASR. It is derived from the Levenshtein distance [38] and calculated as [39, 40]:

$$WER = (\frac{S + I + D}{N}) \times 100. \qquad (18)$$

where S is number of substitutions, I is number of insertions, D is number of deletions, and N is the total number of words in the actual transcript. The interpretation of WER is that the lower the WER, the better the speech recognition is [39, 40].

The loss is also referred to as Expected Transcription Loss [24]. The expected transcription loss function is defined by:

$$\mathcal{L}(x) = \sum_{y} Pr(y|x)\, \mathcal{L}(x, y) \qquad (19)$$

where x is the given input sequence, $Pr(y|x)$ the distribution over transcription sequences y defined by CTC, and $\mathcal{L}(x, y)$ a real-valued transcription loss function.

The edit distance can be best explained with an example. Let us say the normalized edit distance between two words/strings (consider A and B) is $d(A, B)$ [41]. The mean edit distance is calculated by:

$$d(A, B) = min(\frac{W(P)}{N}) \qquad (20)$$

where P is the editing path between string A and string B, $W(P)$ is the total sum of weights of all the edited operations of P, and N is the total number of edited operations (the length of editing path, P) [41].

4.3 Parameter Setup

The following parameters were used for the runs:

- Dropout rate = 30%
- Number of epochs = 10
- Training batch size = 16
- Test batch size = 8
- Activation function = ReLU
- Neuron count in hidden layers = 500 or 1,000 (as indicated)
- Adam optimizer:
  - $\beta_1 = 0.9$
  - $\beta_2 = 0.999$
  - ϵ = 1e-8
  - learning rate = 0.0001

4.4 Results

We ran the three models (RNN, LSTM, GRU) with two different architectures. The first used 500 nodes in each hidden layer whereas the second consisted of a 1,000-node architecture.

Table 1 provides the results of the 500-node architecture experiments. In terms of WER, RNN achieved 87.02%, LSTM 77.55%, and GRU 79.39% with LSTM scoring best. The loss is measured as part of the neural network optimization and shows a similar trend to the WER with 186.61, 160.51, and 162.22 for RNN, LSTM, and GRU, respectively. In terms of the mean edit distance, LSTM achieved the best value with 0.3853.

Table 2 lists the results of the experiments using the 1,000-node architecture. Again, LSTM obtained the best WER value of 65.04% followed by GRU with 67.42% and RNN with 78.66%. The loss values of LSTM and GRU are close with 134.35 and 136.89, respectively. The best mean edit distance is 0.3222 achieved by LSTM.

Figure 2 plots the WER values graphically. The figure clearly shows the lowest WER values achieved by LSTM for both network architectures (500- and 1,000-node network).

Figure 3 shows the WER values of each model (RNN, LSTM, GRU) for the 1,000-node architecture. The test set was applied after each epoch had elapsed, i.e., the model obtained during training is tested on the test data immediately after each epoch and the WER is documented. We can see that LSTM achieves the best WER value as mentioned above. However, what can be observed from the figure is that the best WER value of each model is actually obtained after the 9th epoch. The values at the 9th epoch are 78.43%, 64.76%, 67.34% for RNN, LSTM, and GRU, respectively.

In terms of the running time, RNN beats both LSTM and GRU with the shortest execution times as shown in Figure 4. However, since the WER values of LSTM and GRU are much better, RNN is not really comparable. For the 500-node architecture LSTM ran for more than 2 days whereas GRU ran for approximately 1.5 days. The difference is more significant for the 1,000-node architecture. LSTM ran slightly longer than 7 days whereas GRU only ran for 5 days and 5 hours.

5 Conclusion

Since standard feedforward neural networks cannot handle speech data well (due to lacking a way to feed information from a later layer back to an earlier layer), thus, RNNs have been introduced to take the temporal dependencies of speech data into account. Furthermore, RNNs cannot handle

Table 1: Results for 500-node layer architecture

Model	WER (%)	Loss	Mean edit distance
RNN	87.02	186.61	0.4484
LSTM	77.55	160.51	0.3853
GRU	79.39	162.22	0.3939

Table 2: Results for 1,000-node layer architecture

Model	WER (%)	Loss	Mean edit distance
RNN	78.66	164.60	0.3991
LSTM	65.04	134.35	0.3222
GRU	67.42	136.89	0.3308

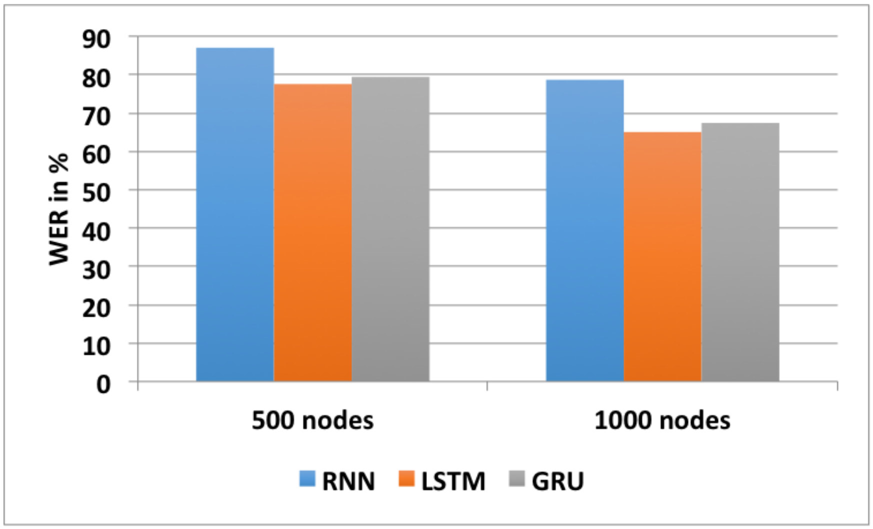

Figure 2: WER results in %

the long-term dependencies due to vanishing/exploding gradient problem very well. Therefore, LSTMs and a few years later GRUs were introduced to overcome the shortcomings of RNNs.

This paper evaluated RNN, LSTM and GRU and compared their performances on a reduced TED-LIUM speech data set. Two different architectures were evaluated; a network with 500 nodes and a network with 1,000 nodes in each layer. The evaluation measures used were WER, loss, mean edit distance, and the running time. The results show that the WER value of LSTM and GRU are close (LSTM scoring slightly better than GRU), however, the running time of LSTM

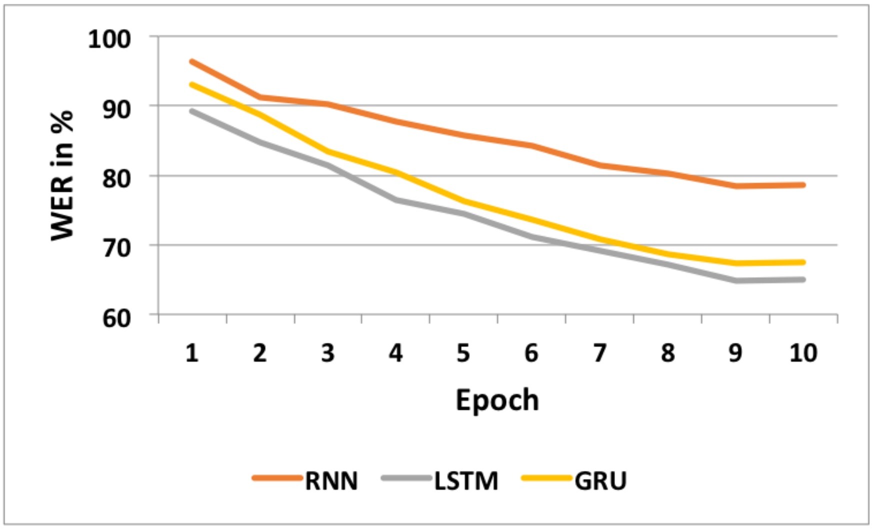

Figure 3: WER values in % per epoch for all models and the 1,000-node architecture

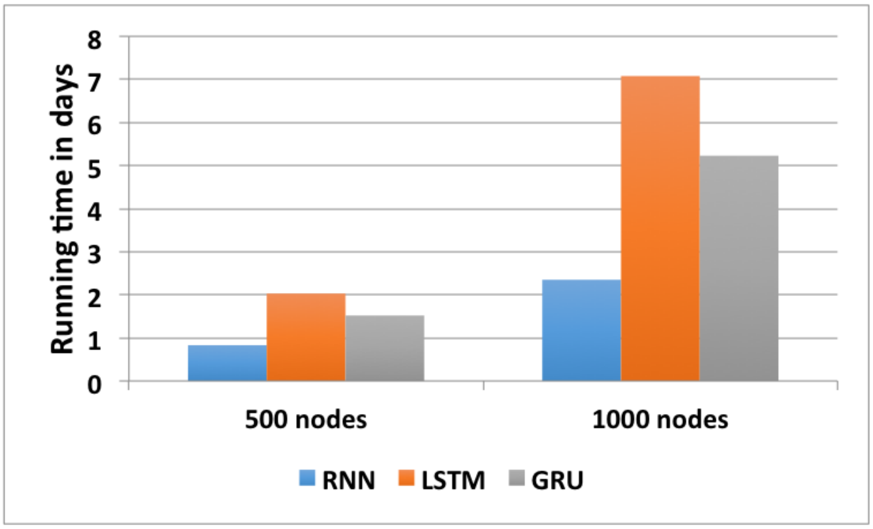

Figure 4: Running time in days

is larger than GRU. Thus, the recommendation for the reduced TED-LIUM speech data set is to use GRU since it returned good WER values within an acceptable running time.

Future work will include parameter optimization in order to investigate the influence on different parameter settings. Furthermore, the learning rate, dropout rate as well as higher numbers of neurons in the hidden layers will be experimented with.

Acknowledgment

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. We also gratefully acknowledge the support of NVIDIA Corporation.

References

[1] G. E. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18, 1527-1554, 2006.

[2] A. Rousseau, P. Delglise, Y. Estve, Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. Proceedings of Sventh Language Resources and Evaluation Conference, 3935-3939, May 2014.

[3] Y. Gaur, F. Metze, J. P. Bigham, Manipulating Word Lattices to Incorporate Human Corrections, Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 2016.

[4] E. Busseti, I. Osband, S. Wong, Deep Learning for Time Series Modeling, Seminar on Collaborative Intelligence in the TU Kaiserslautern, Germany, June 2012.

[5] Deep Learning for Sequential Data - Part V: Handling Long Term Temporal Dependencies, `https://prateekvjoshi.com/2016/05/31/deep-learning-for-sequential-data-part-v-handling-long-term-temporal-dependencies/`, last retrieved July 2017.

[6] Understanding LSTM Networks, `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, last retrieved July 2017.

[7] A. Graves, A. R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6645-6649, 2013.

[8] A. Graves, S. Fernndez, F. Gomez, J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. Proceedings of the 23rd International Conference on Machine Learning, 369-376, ACM, June 2006.

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

[10] TED-LIUM Corpus, `http://www-lium.univ-lemans.fr/en/content/ted-lium-corpus`, last retrieved July 2017.

[11] C. C. Chiu, D. Lawson, Y. Luo, G.Tucker, K. Swersky, I. Sutskever, N. Jaitly, An online sequence-to-sequence model for noisy speech recognition, arXiv preprint arXiv:1706.06428, 2017.

[12] T. Hori, S. Watanabe, Y. Zhang, W. Chan, Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM, arXiv preprint arXiv:1706.02737, 2017.

[13] W. Chan, N. Jaitly, Q. V. Le, O. Vinyals, Listen, attend and spell. arXiv preprint arXiv:1508.01211, 2015.

[14] T. Mikolov, Statistical language models based on neural networks, PhD thesis, Brno University of Technology, 2012.

[15] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014.

[16] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, 3104-3112, 2014.

[17] F. A. Gers, E. Schmidhuber, LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, 12(6), 1333-1340, 2001.

[18] O. Vinyals, S. V. Ravuri, D. Povey, Revisiting recurrent neural networks for robust ASR. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4085-4088, 2012.

[19] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, A. Y. Ng, Recurrent neural networks for noise reduction in robust ASR. Thirteenth Annual Conference of the International Speech Communication Association, 2012.

[20] H. Sak, A. Senior, F. Beaufays, Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv:1402.1128, 2014.

[21] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5), 602-610, 2005.

[22] A. Graves, S. Fernndez, J. Schmidhuber, Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In: Duch W., Kacprzyk J., Oja E., Zadrony S. (eds) Artificial Neural Networks: Formal Models and Their Applications ICANN, Lecture Notes in Computer Science, vol. 3697, Springer, Berlin, Heidelberg, 2005.

[23] A. Graves, M. Liwicki, S. Fernndez, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5), 855-868, 2009.

[24] A. Graves, N. Jaitly, Towards end-to-end speech recognition with recurrent neural networks. Proceedings of the 31st International Conference on Machine Learning (ICML-14), 1764-1772, 2014.

[25] A. Graves, N. Jaitly, A. R. Mohamed, Hybrid speech recognition with deep bidirectional LSTM. 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 273-278, December 2013.

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

[27] H. Xu, G. Chen, D. Povey, S. Khudanpur, Modeling phonetic context with non-random forests for speech recognition. Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[28] T. Ko, V. Peddinti, D. Povey, S. Khudanpur, Audio augmentation for speech recognition. Sixteenth Annual Conference of the International Speech Communication Association, 3586-3589, 2015.

[29] G. Chen, H. Xu, M. Wu, D. Povey, S. Khudanpur, Pronunciation and silence probability modeling for ASR. Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[30] Y. Gaur, F. Metze, J. P. Bigham, Manipulating Word Lattices to Incorporate Human Corrections. Seventeenth Annual Conference of the International Speech Communication Association, 3062-3065, 2016.

[31] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014.

[32] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, End-to-end attention-based large vocabulary speech recognition. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4945-4949, March 2016.

[33] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory. Neural Comput. 9, 8, 1735-1780, November 1997.

[34] K. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, Technical report, arXiv preprint arXiv:1409.0473, 2014.

[35] D. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[36] Reduced TED-LIUM release 2 corpus (11.7 GB), `http://www.cs.ndsu.nodak.edu/~siludwig/data/TEDLIUM_release2.zip`, last retrieved July 2017.

[37] Speech recognition performance, `https://en.wikipedia.org/wiki/Speech_recognition#Performance`, last retrieved July 2017.

[38] Levenshtein distance, `https://en.wikipedia.org/wiki/Levenshtein_distance`, last retrieved July 2017.

[39] A. C. Morris, V. Maier, P. Green, From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. Eighth International Conference on Spoken Language Processing, 2004.

[40] Word error rate, `https://en.wikipedia.org/wiki/Word_error_rate`, last retrieved July 2017.

[41] A. Marzal, E. Vidal, Computation of normalized edit distance and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(9), 926-932, 1993.