Recurrent Neural Network based Language Modeling for Punjabi ASR

*Vaibhav Kumar[1]

[1]*Department of Computer Science & Engineering, Chitkara University Institute of Engineering & Technology, Chitkara University, Rajpura, Punjab, India*

***Abstract* — *Deep Learning approaches have been widely known to perform better than statistical approaches. This is the first effort to investigate Recurrent Neural Network-based modeling for Punjabi speech corpus. We propose the Lattice Rescoring based RNNLM approach using the Kaldi toolkit. Experiments on single sentences showed that the Neural network-based approach performs better than n-gram based modeling approaches. A performance improvement of 7-9% on word error rate (WER) was observed on top of the state-of-the-art Punjabi speech recognition system.*

Index Terms — *automatic speech recognition, recurrent neural network language modeling, lattice rescoring, Punjabi ASR*

I. INTRODUCTION

Punjabi is a popular and widely spoken language worldwide. But in the speech recognition system, Punjabi fails to compete with other languages due to the lack of research in the Punjabi language for ASR. Language modeling has been a critical aspect of Automatic Speech Recognition (ASR). With the advancements in computational power, neural network approaches had grown by leaps and bounds, and so does the experiments with those approaches in the field of ASR. Over the times, the traditional n-gram approaches had been used for language modeling, but with RNNLM coming into the picture, the ASR system's performance had been quite impressive. RNNLM is known to have a history vector associated with it, which helps keep track of the previous state of the sentence.

The language model is a probability distribution that tries to learn the context between different words and phrases. While dealing with words, data variability is an issue with building language models. Traditionally, it is believed that the probability of a current word depends on previous n-words. That is known as the n-gram language model. But in order to learn the distribution and context between words and generalizing the relationship better between words, deep learning-based approaches had been proven to be a handful. So, RNN while Trying to predict the probability keeps in mind the history vector, which means it has the previous context and relationships and thus generalizes better.

Statistical modeling fails to generalize well, and it has often been seen with computer vision as well as speech in various languages that deep learning-based models tend to perform well compared to statistical models. This urges to conduct experiments with RNN based language models for the Punjabi language. With the increasing amount of data, deep learning models tend to benefit from those data. But simple deep neural networks are unable to keep track of the history; rather, they focused on the current word of the sentence. Hence RNNLM, which has a history vector, tends to perform better than DNN and n-gram modeling approaches.

Researches have been going on for the Punjabi language ASR with the experiments being conducted with deep learning methods. Researchers have been trying to improve lattice rescoring methods. In [3], RBM based unsupervised learning with the DNN-HMM system has been explored, and results have been compared to GMM-HMM modeling.

In [6], authors investigated the potential of context-dependent DNN-HMM with a huge amount of dataset by applying feature transforms for VTLN, fMLLR applied to CD-DNN-HMM setup. Results showed that there was again using fMLLR based transforms. In [11], for Punjabi language DNN-HMM system has been evaluated with different monophone and triphone training. Alongside MFCC and GFCC, features have been used, which results in improvement for the performance. In [2], RNNLM had been used. DNN adaptation models for RNNLM have been used, which has resulted in WER improvements. In [14], researchers have concluded that deeper structures can result in improvement for the network. In [7], the authors explored the recent advances in language modeling using RNN's. Alongside a study has been performed on Convolutional neural networks (CNN's) and Long-short term memory (LSTM's) over a sufficiently larger dataset. In [4], RNNLM is being trained on a larger dataset, which concludes the improvements with increasing data for this approach. RNNLM tends to outperform n-gram language modeling with an increased amount of data.

In [5], the Lattice generation method is being used for on-the-fly lattice rescoring for real-time ASR system that gives a lot of intuition about how crucial is lattice rescoring while processing audio in real-time. The different systems used gave accuracy improvement for on-the-fly rescoring without adding too much latency, which is crucial while performing real-time decoding of input audio.

In the remainder of this paper, key ideas behind the Punjabi language are introduced. In Section 2, the basics of Recurrent Neural Networks are being described, followed by exploring lattice rescoring (Section 3). Concluding with the experimental setup in Section 4 and experimental results in Section 5, discussion on future work is being carried out in Section 6.

In our work, we focused on generating a lattice rescoring based approach for RNNLM, which helps in generating the lattice that can keep track of history vector, which is believed to give better results for ASR having word error rate as a metric for measuring the performance of the ASR system. The language models have been evaluated with different acoustic modeling approaches discussed in later sections, with the input signal being converted into MFCC features. For training and evaluation of this approach for Punjabi corpora, an open-source automatic speech recognition toolkit Kaldi has been used. Using Kaldi and Tensorflow, the RNNLM based approach is implemented and evaluated. Also, the proposed approach is being compared with an n-gram modeling approach.

II. RECURRENT NEURAL NETWORK MODEL

RNN remembers the past, and its predictions and outcomes are based on the things it had learned from the past. Whereas the feed-forward network remembers only the things it learned during training, but the recurrent neural network remembers the information gathered from prior inputs. That's why they have been proven to perform better in the case of sequence learnings where posterior output is based on the current input as well as the prior input.

Every layer in a deeper MLP has its own weights. So, there is no correlation between weights learned in an earlier layer. So RNN, in such a case, proves to be a handful where context mapping and learning is required.

RNN can produce one or more outputs determined not only by the weights applied but also by the hidden state vector that represents context based on the previous inputs/outputs. So, courtesy of this context-based learning, the same sequence can have different predictions depending upon the prior inputs in the sequence of inputs.

In Figure 1., the Sequence of inputs $I_n = \{I_1 , I_2 , I_3 , \ldots , I_n\}$ is fed as input to the network having input weight matrix as $W_n = \{W_1 , W_2 , W_3 , \ldots , W_n\}$ along with hidden state $H_n = \{H_1 , H_2 , H_3 , \ldots , H_n\}$. There can be any number of hidden layers having hidden units used for context mapping and history learning, which help in outputting the sequence using weights associated with hidden layers as $O_n = \{O_1, O_2, O3, \ldots, O_n\}$. The history vector is carried along with the hidden units, which is the reason for the better performance of RNN over other DNN based networks for tasks like language modeling. Those hidden nodes preserve the sequential information in the case of recurrent networks and cascade forward to affect the processing of each new example. This information is used to perform tasks that the feed-forward networks can't.

The number of hidden layers in a network is a hyperparameter that needs to be tuned according to the behavior of the network with the provided dataset. Experiments can be conducted to find the optimum number of hidden layers and hidden units. In the past, it has been observed that a smaller number of layers can perform better than a larger number of layers on a small dataset. But having large variability in data, the model might need to be bigger to learn complex features.

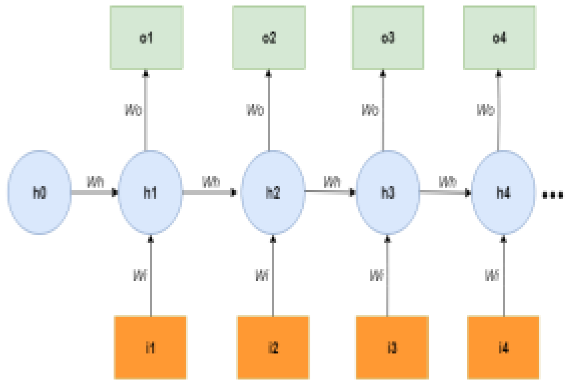

Figure 1: *Simple RNN model architecture with input, hidden, and output layer.*

For calculating the current state, the following formula is used: -

$$h_t = f(h_{t-1}, x_t) \quad (1)$$

where h_t represents the current state of being calculated using previous state h_{t-1} and input x_t .

After getting the state, activation needs to be applied using the formula: -

$$h_t = \mathrm{ReLU}(W_{hh} h_{t-1} + W_{xh} X_t) \quad (2)$$

where ReLU is the activation function used with W_{hh}

being the weight of recurrent neuron and W_{xh} weight at input neurons.

The activation function plays a key role in making sense of complicated and non-linear complexities in the data and decides whether a neuron should be activated or not. They introduce non-linearities in the network. After passing through the activation layer, the signal becomes ready to be fed into the next layer. Experiments had been concluded in order to find a suitable activation function, which is described in detail in Section 5.

III. LATTICE RESCORING

A Lattice is interpreted as a weighted, labeled, and directed acyclic graph generated using single knowledge sources. Lattices have a path for every word from the vocabulary in the hypothesis within the limits of the search beam. Lattice rescoring involves two kinds of floating points, one being the graph cost and the other being the acoustic cost. To do achieve things like lattice rescoring, primarily, the old LM cost is subtracted, and then the new LM cost is added. In other words, Lattice rescoring refers to the second pass of speech decoding, which is done with lattices generated in the first pass using high-level knowledge resources such as the n-gram language model and, in our case, RNN based language models. There is a tradeoff between accuracy and speed when choosing between the n-gram language model and RNNLM based language model. Lattice rescoring using RNNLM tends to be more accurate, so we had decided to go with accuracy over speed. So, as a result of modeling using RNNLM, a sequence can have different paths and, as a result, different outputs due to the history vector and prior input sequences.

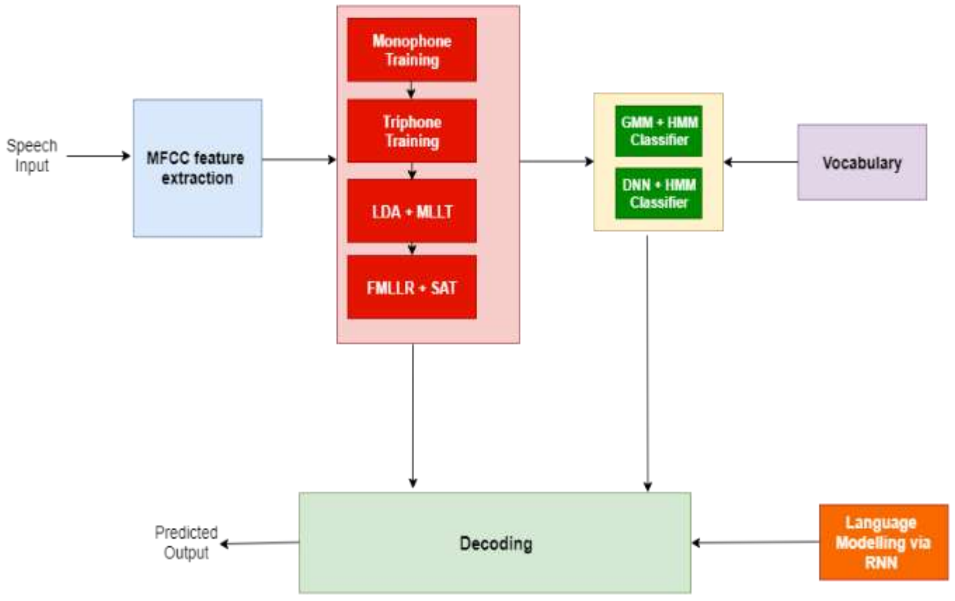

Figure 2: *Block Diagram indicating the system that is being used for training the ASR.*

IV. EXPERIMENTAL SETUP

A. Dataset Collection and Distribution

The Proposed system has been trained and evaluated with different combinations of acoustic and language modeling setups using extracted feature vectors. The dataset consists of recordings from various female and male speakers from different regions of Punjab. The dataset involves both good and bad recording samples. Bad recording example means noisy data, which helps the system learn the kind of data that it might receive during real-time testing and on-the-fly predicting transcription for an utterance. The average sentence length is around 10 words. For recording purposes, Dictaphone had been used. The dataset has been divided into two parts, namely training and validation. Training consists of 85% of data, while the rest 15% had been used for validation. Word Error Rate % (WER) has been used as a metric to evaluate the performance alongside perplexity, which helps to determine to what extent RNNLM has learned the context mapping for distributions of the dataset.

B. System Overview

The shape of the vocal tract determines the sound produced by humans. If this shape can be determined accurately, the sound produced can be determined correctly. So, Mel Frequency cepstral coefficients (MFCC) helps to determine the envelope of the time power spectrum of the speech signal. Due to this behavior of MFCC features, this technique is being used to extract a feature vector for creating an ASR system. These MFCC features act as input to the model. The entire system in Figure 2 uses these feature vectors using monophone training (M1), triphone training using Delta and Delta+Delta (Tri1), triphone with LDA + MLLT (Tri2), and triphone with FMLLLR + SAT (Tri3) with classifiers being GMM+HMM and DNN+HMM.

Initially, monophone training is done, and lattices are stored. Then those lattices are fed as input for triphone training using Delta and Delta+Delta, and then lattices outputted from these are saved to the disk and then used at the next step. This goes on until the last step [11].

Vocabulary is being fed into the classifier, which consists of all the unique words found in the training transcription. Language modeling is being done via RNN model training. Alongside 3-gram and 4-gram language modeling is also done to see what kind of improvements is shown by the recurrent network over the traditional n-gram language model. The Learning rate decay has been used to ensure the step that has been taken during gradient descent is controlled. If we set the learning rate too small, then the network might not learn, and if set too large, we may overshoot the area of low loss or start overfitting. So, considering this importance of learning rate, for RNNLM to not overfit the data as well as learn the complex mapping learning rate, decay is essential.

C. *System overview for DNN-HMM model*

Deep neural network (DNN) is known for its learning ability and generalizing ability for the provided input features, and HMM is known for its sequential modeling. But DNN alone won't be helpful as it would take only fixed-size input. So, for this case, HMM comes into practice, which helps in taking dynamic input, and DNN helps in learning complex things. The generated lattices from other triphone techniques had been used with the DNN-HMM classifier. Experimental parameters for the system were: -

Hyper-parameter	**Value**
Learning rate	0.005
Number of hidden layers	3
Number of hidden units	512
Mini batch size	128

Table 1: *Value of hyper-parameters used for the DNN-HMM classifier.*

So, the hyper-parameter needs tuning. The learning rate had been varied from 0.5 to 0.00001, but 0.005 had been found optimum. The number of hidden layers was varied from 3-5 but have not found much difference in the performance of the model. This has been kept to 3 alongside other parameters that hadn't been tinkered much for this classifier. So other parameters had been kept to their default values.

V. EXPERIMENTAL RESULTS

A. *Effect of various activation functions*

The Hyperbolic tangent (tanh) and the ReLU activation function had been known to perform well with deep learning approaches.

The tanh activation function is: -

$$f(x) = \frac{2}{(1 + e^{-2x})} - 1$$

and the ReLU activation function looks like: -

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

These two activation functions have been varied with the RNNLM setup first up to find optimal activation function. Perplexity has been used to see how well the model learns and how the loss converges.

Activation function	**Train Perplexity**	**Valid Perplexity**
Tanh	82.7	85.1
ReLU	66.3	68.4

Table 2: *Values of Perplexity for various activation functions tested.*

After trying various activation functions, the non-linearity function ReLU proves to perform well with the RNNLM setup as expected from theoretical knowledge. Using ReLU as an activation function, perplexity seems to converge well. For carrying out this experiment, the number of hidden layers for RNNLM has been kept to 2 with a mini-batch size being 64. The experiments and tuning of hyper-parameters being quite necessary to have a robust model that would give better results and performance than the baseline model. This setup is done just to find the optimal activation function, whereas other hyper-parameters have been optimized later on, while some have been determined using heuristics or empirically been set to a particular value.

B. *Effect of the number of hidden layers*

While finding the optimal activation function, an intuition is made that the model might not be generalizing well, which was clear from the high perplexity values, i.e., the number of hidden layers being 2 might not be sufficient for the model to learn complex features. So, the number of hidden layers needs to be tuned.

There has been a tradeoff between time and accuracy with the increasing number of hidden layers. Increasing the number of hidden layers will increase the number of trainable parameters for the model. As a result, the model will take a longer time to train.

Figure 3. indicates the variation of training perplexity with the number of hidden layers being 2, 3, and 4. Initially, it has been observed that the number of hidden layers for the initial epoch the perplexity is high, but as the training progress, the model tends to show the behavior as expected. Perplexity values tend to decrease with the increasing number of epochs. The decrease in perplexity being sharper for a higher number of hidden layers, i.e., perplexity decreases more in case of hidden layer 4 as compared to that in hidden layer 3, which means data is being fitted properly. The values for the validation perplexity had shown nearly the same behavior, as shown by training perplexity. Validation perplexity decreases as well with an increased number of epochs, which means training is going well and models in learning patterns from data instead of learning the data. Hence the number of hidden layers for RNNLM being 4 had shown good behavior with respect to training and validation perplexity.

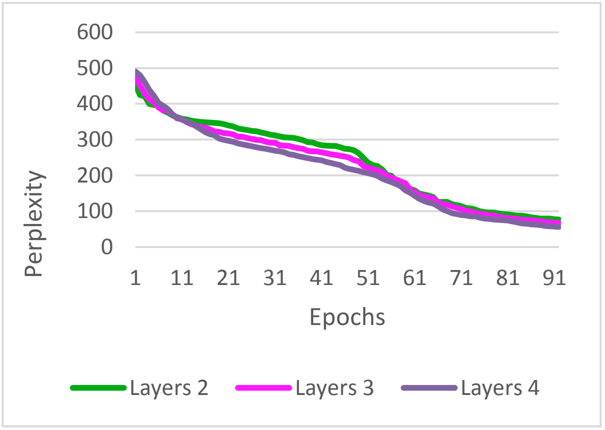

Figure 3: Variation of Training Perplexity with varying number of hidden layers plotted alongside the number of epochs for which the model is being trained.

C. *RNNLM setup parameters*

There is no definite rule of thumb on how many numbers of layers would be there in the network, what should be the number of neurons in the hidden layer etc. Very often, a trial and error approach will give the results regarding such kind of hyper-parameters. After performing all the experiments on the hyper-parameters for RNNLM, optimal parameters had been found. The model tends to do well with the following set of parameters: -

Hyper-parameter	Value
Learning rate	0.7
Number of hidden layers	4
Number of hidden units	256
Mini batch size	64

Table 3: Value of hyper-parameters used for RNNLM setup.

The learning rate and the number of hidden units are not being tinkered much with the focus being on tuning the number of hidden layers to be used and finding the activation function for the current RNNLM setup. Learning rate decay has been used for the gradient to descent to the optimal value. So, after every epoch, the learning rate is being decayed, and for this purpose, only the learning rate has been sat up to a higher value initially.

D. *Comparison with baseline modeling techniques*

The input audio signal is being converted to feature vector and then used for different approaches for language modeling and acoustic modeling. To enhance the performance of ASR, lattice rescoring at various levels of acoustic modeling is being done. 3-gram and 4-gram language modeling is being evaluated along with RNN based language modeling, which is depicted in Table 4.

System Type	Language Modelling		
	3-gram	4-gram	RNNLM
Tri1	36.76	36.39	34.51
Tri2	35.29	34.28	32.85
Tri3	32.03	31.64	28.32
DNN-HMM	29.93	27.83	24.76

Table 4: WER in % for various language modeling techniques being evaluated.

For all the systems evaluated, WER% decreases while going from 3-gram language modeling to RNNLM. But the results have been quite better when the model is being evaluated with DNN-HMM setup for Punjabi corpus. This shows adaptive learning by the deep learning-based language models as compared to the statistical models of Punjabi language. The recurrent neural network-based language model outperformed the baseline n-gram language model, as depicted by the evaluation metric.

VI. CONCLUSION AND FUTURE SCOPE

Conducting experiments with different acoustic modeling with different language modeling approaches helped in finding a good language model. This also made clear that theoretically, it is known that deep learning-based language model generalizes well, which was clear from the results using RNN based language modeling. The approach yielded an improvement of 7-9% with the DNN-HMM system and RNN based language modeling for Punjabi corpora.

Further work can be extending our network used and also exploring other approaches like long-short term memory (LSTM), which is known to perform better than RNNLM. Having a good response to RNNLM experiments can be conducted with similar approaches to enhance the performance of a speech recognition system. Also, in order to have a more robust system, different kinds of data need to be added, like dialectal data, noisy data, etc. As collecting data is quite labor-intensive and deep learning techniques require a large amount of data, so data perturbation techniques can be explored, keeping in mind the future agenda.

Acknowledgments: - This work is partially tested on the sample Punjabi corpus collected for Language Resources for Auditory impaired Person project. The results in work are as per the perspective of the research. The authors would like to thank Speech and Multimodal Laboratory members Puneet, and especially Dr. Virender Kadyan at Chitkara University Punjab. Special thanks to Dr. Banriskhem Khonglah, Researcher at Idiap Research Institute, who helped by providing valuable input for RNNLM.

Funding: - No funding was granted for this project.

Conflict of Interest: - The author declares that they have no conflict of interest.

VII. REFERENCES

[1] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "*A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition*," in Acoustics, Speech, and Signal Processing (ICASSP), 2018 IEEE International Conference on. IEEE, 2018.

[2] Ke Li, Hainan Xu, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, "*Recurrent neural network language model adaptation for conversational speech recognition*," Proc. Interspeech, 2018, pp. 3373–3377, 2018.

[3] L. Longfei, Z. Yong, J. Dongmei, Z. Yanning, W. Fengna, I. Gonzalez, et al., "*Hybrid Deep Neural Network--Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition*," in Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on, 2013, pp. 312-317.

[4] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011. "*RNNLM-recurrent neural network language modeling toolkit*". In Proceedings of the 2011 ASRU Workshop pages 196–201.

[5] H. Sak, M. Saraçlar, and T. Güngör, "*On-the-fly lattice rescoring for real-time automatic speech recognition*," in Proc. Interspeech, 2010, pp. 2450–2453.

[6] F. Seide, G. Li, X. Chen, and D. Yu, "*Feature engineering in context-dependent deep neural networks for conversational speech transcription*," in Proc. IEEE ASRU, 2011, pp. 24–29.

[7] Jozefowicz, Rafal, Vinyals, Oriol, Schuster, Mike, Shazeer, Noam, and Wu, Yonghui. "*Exploring the limits of language modeling*". arXiv preprint arXiv:1602.02410, 2016.

[8] X. Chen, T. Tan, X. Liu, P. Lanchantin, M. Wan, M. J. Gales, and P. C. Woodland, "*Recurrent neural network language model adaptation for multi-genre broadcast speech recognition*," in Sixteenth Annual Conference of the International Speech Communication Association, 2015

[9] M. Bod' en, "*A guide to recurrent neural networks and backpropagation*," in In the Dallas project, SICS Technical Report T2002:03, SICS, 2002.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al., "*The kaldi speech recognition toolkit," in IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFLCONF-192584. IEEE Signal Processing Society, 2011.

[11] V Kadyan, A Mantri, RK Aggarwal, A Singh, "*A comparative study of deep neural network-based Punjabi-ASR system*" in International Journal of Speech Technology 22 (1), 111-119.

[12] H. Sak, Hasim, Senior, Andrew, and Beaufays, Francoise. "*Long short-term memory recurrent neural network architectures for large scale acoustic modeling*". In Interspeech, 2014.

[13] Robinson, Tony, Hochberg, Mike, and Renals, Steve. "*The use of recurrent neural networks in continuous speech recognition".* pp. 253–258, 1996.

[14] N. Morgan, "*Deep and wide: Multiple layers in automatic speech recognition*," IEEE Trans. Audio Speech Lang. Processing, vol. 20, no. 1, Jan. 2012, pp. 7–13.

[15] Parthasarathi, S. H. K., Hoffmeister, B., Matsoukas, S., Mandal, A., Strom, N., & Garimella, S. (2015). "*fMLLR based feature space speaker adaptation of DNN acoustic models*". In the Sixteenth annual conference of the international speech communication association

[16] Sivasankaran, S., Nugraha, A. A., Vincent, E., Morales-Cordovilla, J. A., Dalmia, S., Illina, I., et al. (2015). "*Robust ASR using neural network-based speech enhancement and feature simulation. In IEEE workshop on automatic speech recognition and understanding (ASRU)*", 2015 (pp. 482–489).

[17] T. Mikolov, I. Sutskever, A. Deoras, H. S. Le, S. Kombrink, and J. Cernock' y, "*Compression of Language Models Using Subword N ˇ Neural Networks,"* in Submitted to ICASSP, 2012.

[18] I. Sutskever, J. Martens, and G. Hinton, "*Generating Text with Recurrent Neural Networks*," in Proceedings of ICML, 2011.

[19] Y. Bengio, R. Ducharme, P. Vincent et al., "*A neural probabilistic language model,"* Journal of Machine Learning Research, vol. 3, pp. 1137–1155, 2003.

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

[21] M. Sundermeyer, R. Schluter, and H. Ney, "*Lstm neural networks ¨ for language modeling,*" in Thirteenth Annual Conference of the International Speech Communication Association, 2012.

[22] X. Chen, A. Ragni, X. Liu, and M. J. Gales, "*Investigating bidirectional recurrent neural network language models for speech recognition,*" Proc. ICSA INTERSPEECH, 2017.

[23] Ronald J. Williams and Jing Peng, "*An efficient gradient-based algorithm for online training of recurrent network trajectories*," Neural Computation, vol. 2, pp. 490–501, 1990.

[24] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "*Learning long-term dependencies with gradient descent are difficult*," Neural Networks, IEEE Transactions on, vol. 5, no. 2, pp. 157–166, 1994.

[25] B. Roark, M. Sarac¸lar, and M. Collins, "Discriminative n-gram language modeling," Computer Speech and Language, vol. 21, no. 2, pp. 373–392, April 2007

[26] S. Mittal, R. Kaur, "Implementation of phonetic level speech recognition system for Punjabi language", *2016 1st India International Conference on Information Processing (IICIP)*, pp. 1-6, 2016.

[27] Prashanth Kannadaguli and Vidya Bhat, "Paper Title" SSRG International Journal of Electronics and Communication Engineering 1.9 (2014): 1-4.