IDIAP RESEARCH REPORT

INVESTIGATING TIME DELAY NEURAL NETWORK (TDNN) FOR LANGUAGE MODELING IN LOW RESOURCE AUTOMATIC SPEECH RECOGNITION

Banriskhem Khonglah Srikanth Madikeri
Petr Motlicek Hervé Bourlard

Idiap-RR-13-2019

OCTOBER 2019

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

INVESTIGATING TIME DELAY NEURAL NETWORK (TDNN) FOR LANGUAGE MODELING IN LOW RESOURCE AUTOMATIC SPEECH RECOGNITION

Banriskhem K. Khonglah, Srikanth Madikeri, Petr Motlicek, Hervé Bourlard

Idiap Research Institute, Martigny, Switzerland
{banriskhem.khonglah, srikanth.madikeri, herve.bourlard, petr.motlicek}@idiap.ch

ABSTRACT

This work discusses the potential of using Time Delay Neural Networks (TDNN) for language modeling in automatic speech recognition (ASR) of low resource languages. Recently, TDNN for language modeling has been shown to achieve better perplexity than an equivalent Recurrent Neural Networks (RNN). However, its potential for ASR decoding has not been explored yet. In this work, we focus on exploring the strength of TDNN as a language model for ASR in low resource languages. TDNNs are significantly easier to train than RNN since they have a feedforward structure instead of a recurrent structure. We explore a TDNN-based language model both directly in the decoding graph and as well as in the rescoring phase. Experiments show that the TDNN LMs perform as the RNN, although it is faster and easier to train the TDNN compared to RNN. Two languages from the BABEL corpora are considered in this work: Tagalog and Swahili.

Index Terms— recurrent neural network, time delay neural network, low resource, language modeling

1. INTRODUCTION

In automatic speech recognition (ASR) systems, neural based language models are often used to rescore top scoring hypotheses obtained using conventional back-off n-gram language models (LM). Alternately, the lattices generated with n-gram LMs may be directly rescored. Recurrent Neural Networks (RNN) are the most commonly employed neural LMs in ASR systems [1–5].

Recently, a LM based on Time Delay Neural Networks (TDNN), in which the convolution is applied with respect to only the past time steps to avoid any leakage from the future time steps, was proposed [6]. The TDNN LM was shown to have lower perplexity than a RNN LM. It was also mentioned in [6] that the TDNNs should be used as a starting point for various sequence modeling task. However, whether TDNNs can be used for ASR was not discussed. Hence this work focusses on exploring this, specifically in terms of ASR and if TDNNs can be used over RNNs.

There are various reasons why the TDNN is preferred over RNNs for sequence modeling task [6]. In TDNNs, the long sequence as a whole is processed. This means that the parallel convolutions can be done since the same filter is used in a particular layer. In RNNs, there is some sort of sequential processing, since the predictions in the next time steps depend on the predecessors to complete. TDNNs do not suffer from the vanishing gradient problem which is common in RNNs. The receptive field size of a TDNN is better compared to the RNN. This gives better control of the memory size of the model, and it can also be used to adapt to different domains very easily.

The training of the TDNN in our work is almost similar to the work in [6]. However, in that work, it was not tested for ASR. In order to use the network for ASR, some changes to the original training have to be incorporated. The sentences were processed in a dependent manner in [6] and the perplexity obtained seemed promising. In our work, the sentences are processed independently and not only is the perplexity computed, but also word error rate which is useful for evaluating the ASR performance.

A neural LM such as TDNN or RNN can be used for decoding in ASR. This can be in terms of the first pass decoding or the second pass decoding. The first pass decoding involves using the neural LM probabilities for graph creation. There are a few examples in the literature for first pass decoding. In [7], the RNN was used as a generative model to generate text. The n-gram LMs are then trained based on the generated text and finally used for decoding. In [8], RNN LM histories are discretized to create weighted finite state transducers (WFST). A probability based conversion has been explored in [9], and this method involves the extraction of n-grams but the count based probabilities are replaced by the RNN LM probabilities. The second pass decoding involves the rescoring of the hypothesis generated from the first pass decoding by an n-gram LM [1].

The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via AFRL Contract FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

In this work, the TDNN is explored for creating the Language Model to be used for decoding in ASR for low resource languages and this neural based LM will be used in both the first pass decoding and the second pass decoding. For the first pass decoding, the probabilities of the n-grams generated by simple counting, will be replaced by the probabilities estimated by the TDNN. The TDNN will also be used to generate probabilities for the n-grams computed from the N-best list. The use of n-gram probability generation is much clearer using TDNN since we know the history represents the previous words. On the other hand, the RNN has a continuous history [8] represented by the recurrent layer and hence it is not very clear if the history is actually represented just by the previous words. Hence estimating the n-gram probabilities by the TDNN gives a clearer picture than by using the RNN as done in [9]. The probabilities generated by the two will be explored for the decoding process.

For the second pass decoding, the standard method [1, 2], will be used with the probability assignment of the sentences being computed by the TDNN. The performance of the TDNN will be compared to the standard n-grams as well as the RNN LM using first and second pass decoding, in terms of perplexity, word error rate and computational speed. The specific contributions of this paper are as follows:

- Exploring TDNN for decoding as a language model in ASR tasks for low resource languages (Tagalog and Swahili)
- Using TDNN for second pass decoding
- Replacing probabilities of the n-grams generated by simple counting with TDNN probabilities expecting that the TDNN will be able to model the probabilities better

The rest of the paper is organised as follows. Section 2 discusses the RNN LM. Section 3 introduces the TDNN. Section 4 discusses the ASR decoding while section 5 describes the acoustic models. The experiments are given in section 6. Finally the conclusion is given in section 7.

2. RECURRENT NEURAL NETWORK LANGUAGE MODELS

The recurrent neural network [1, 2] used for comparison in this work is shown in Fig. 1. It consists of an input layer that reads a one-hot encoding representing each previous word I(t), and the previous state of the hidden layer C(t-1). The hidden layer compresses the input information and produces a new state C(t). C(t) is then passed to the output layer which has the same dimension as I(t). The output layer produces the conditional probability. This is the probability of the next word given the previous word and the previous state of the hidden layer. Training is done using the stochastic gradient descent algorithm and the weights are trained using the truncated backpropagation through time (BPTT) algorithm [1]. There is also an additional class layer at the ouput [2,3] which helps to reduce the computational complexity at a small cost of accuracy.

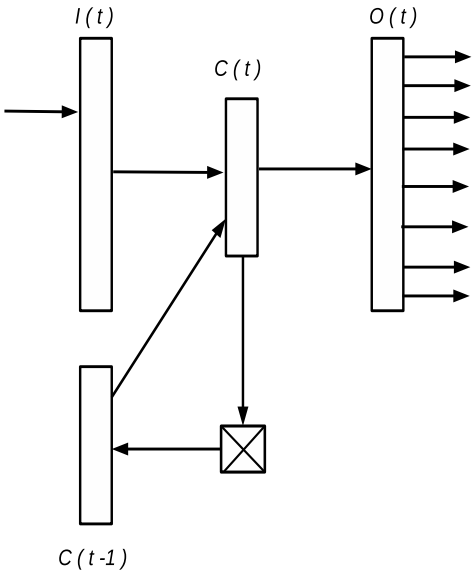

Fig. 1. *Architectural elements of an RNN with a single hidden layer*

3. TIME DELAY NEURAL NETWORKS

The time delay neural network (TDNN), show in Fig. 2, was proposed in [6] for various sequence modeling tasks. Unlike conventional TDNNs, we employ a causal version where there is no leakage of information from the future time steps. That is, the output at a particular time is convolved only with elements from that time and earlier in the previous layer. Also the size of the input and output is the same, which means that the zero padding is added to keep the next layers same as the previous ones. It is also called as a temporal convolutional network (TCN) in [6]. This architecture resembles a finite impulse response (FIR) implementation, unlike the RNN which is analogous to an IIR filter. A similar architecture was proposed in [10], where the only difference is the padding of zeros to make sure the input is equal to the output. It had the disadvantage that an extremely deep network is required to achieve a long effective history. On the other hand, the TDNN in our work can be used to incorporate a long effective history by using a combination of deep networks and dilated convolutions as follows:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i).x_{s-d.i} \qquad (1)$$

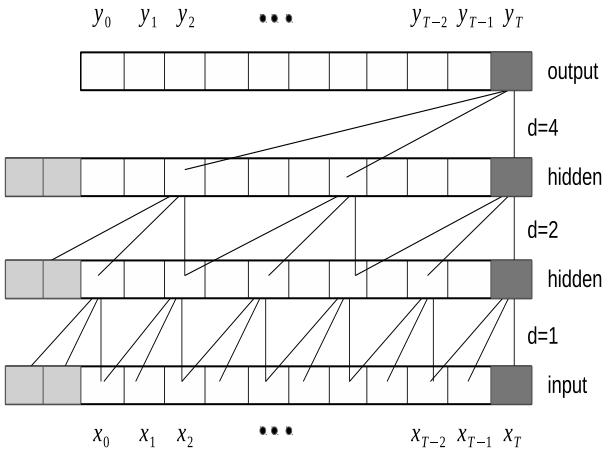

Fig. 2. *Architectural elements of a TDNN, with filter size of 3 and having several dilations in different layers*

where d is the dilation factor, k is the filter size, and $s - d.i$ accounts for the direction of the past. The receptive field of the TDNN can be increased by increasing the filter size and using larger dilations d where the effective history is $(k-1)d$.

4. ASR DECODING

4.1. First Pass Decoding

The first pass decoding being implemented in this work is based on replacing the n-gram probabilities with the probabilities of the TDNN and then creating a finite state transducer with the new n-gram probabilities. Another method of first pass decoding that will be used will be based on the N-best list extracted from the lattices generated by the n-grams. The N-best list are used to generate the new n-grams and the probabilities of the new n-grams are estimated by the TDNN instead of the normal counting. The similar process will be followed using RNN for comparison.

4.2. Second Pass Decoding

In this form of decoding, the TDNN will be used to rescore the N-best hypothesis generated from the lattices constructed using n-gram in the initial pass. The scores are computed as follows,

$$P_x(w_i|h) = \lambda P_{tdnn}(w_i|h) + (1-\lambda)P_{ng}(w_i|h) \quad (2)$$

where w_i is the word, h is the history, P_{tdnn} is the probability estimated by TDNN, P_{ng} is the probability estimated by the n-gram and P_x is the combined probability estimate.

The log likelihood score for each N-best hypothesis is then,

$$logL(s) = n.wp + \sum_{i=1}^{n} asc_i + lms \sum_{i=1}^{n} logP_x(w_i|h_i) \quad (3)$$

where, wp-word insertion penalty, asc-acoustic score, n-number of words and lms-language model scale

5. ACOUSTIC MODELLING

Two acoustic model architectures are used in this paper: (i) a typical DNN/HMM system with the DNN trained using a cross-entropy loss function, and (ii) a lattice-free MMI (LF-MMI) based DNN/HMM system. Both acoustic models are implemented in Kaldi. The systems are trained using 16 dimensional Perceptual Linear Prediction (PLP) features appended with 3 dimensional pitch features. The features are mean normalized over each audio channel. Speech/non-speech activity is A conventional HMM/GMM system is trained with these features using the standard Feature space Maximum Likelihood Linear Regression (fMLLR) based speaker adaptive training (SAT).

Alignments obtained from the HMM/GMM system are used as outputs to the DNN/HMM system. A DNN with 4 hidden layers and p-norm activations are used. An initial learning rate of 0.0015 and a final learning rate of 0.00015 are used to train the DNN with Stochastic Gradient Descent (SGD). The learning rate is halved when the frame accuracy on the validation set fails to improve. In addition, sequence discriminative training is applied with the state-level minimum Bayes risk criterion with a low learning rate of 0.00001.

LF-MMI models are trained using features at a frame rate of 30 Hz. MFCC features with 40-dimensions are used to train the system. In place of SAT, online i-vectors are used to train the system and decode on test data in order to retain the advantageous of LF-MMI in terms of speed. A TDNN with 7 layers is trained with each layer having 1024 rectified linear units. LF-MMI models are trained with an initial learning rate 0.001 and a final learning rate of 0.0001 with a momentum of 0.1 for 4 epochs. Cross-entropy and leaky HMM regularization are applied. The latter is applied with a co-efficient of 0.1.

6. EXPERIMENTS

The experiments are performed on datasets of two languages namely Tagalog and Swahili. The babel set is used for the experiments. For Tagalog there were 93104 sentences (594809 words) for training and 917 sentences (5079 words) for the development set. This development set was used to compute the perplexity of the various models. There were 39354 sentences (250398 words) for training the Swahili language model and 1249 sentences (6889 words) were used for development.

Table 1. **Perplexity on the Babel Set of Tagalog and Swahili**

	Perplexity	
Language Model↓	Tagalog	Swahili
n-gram	124.1	137.2
RNN	104.5	129.1
TDNN	98.5	119.6

The TDNN implemented was the one found in [6], along with some modifications in the original implementation. The main modification in the original TDNN is processing the sentences independently for both training and testing as is done for the RNN training. The number of hidden layers was taken to be 2. The size of the word embeddings is fixed to 600 and the number of hidden nodes are also taken to be 600. The kernel size was taken to be 3. The RNN toolkit [11] was used for the RNN training, which also trains the sentences independently. For the n-grams, the SRILM toolkit was used. The n-gram used in the experiments is the 3-gram without any pruning. The acoustic models were trained using the Kaldi toolkit [12].

6.1. Perplexity

Initially the experiments are performed by computing the perplexity of the TDNN and comparing it with the perplexities of the n-gram and RNN LM models. The respective models are trained using the babel text for Tagalog and Swahili. The training samples are passed in a random manner. The vocabulary size is taken to be 20k words. The out of vocabulary words are mapped to a 'unk' symbol. The results are shown in Table 1. It can be observed that the perplexity of TDNN is lower than n-gram LM and RNN LM on both Tagalog and Swahili. This shows that the TDNN is able to model the context better compared to both RNN and n-gram and this has also been shown in [6], where it is mentioned that TDNN exhibits substantially longer memory.

The TDNN has shown better perplexities compared to RNN and n-gram. However their ability to perform on the decoding tasks of ASR has not been explored yet. In ASR tasks, the length of the sentences are not so long and it will be interesting to see if the TDNN is able to improve the performances on ASR as the RNN LM. The ASR task considered in this work is on low resource languages such as Tagalog and Swahili.

6.2. Word Error Rate (WER) for second pass decoding

The most common way of evaluating the performance of a neural language model on the decoding task of ASR systems is by the second pass rescoring. Initially the decoding is done using the n-gram LMs. Once the lattices are formed, they are used to extract the N-best list. The N value can be increased to get good performances while at the same time maintaining a faster decoding speed. The N value taken in this work is 1000. This value is taken according to the best accuracy without compromising the decoding speed. The N-best list created are then passed through the TDNN and the probability of the whole sentence is computed. This probability is then used as described in section 4.2. The results using the second pass decoding are shown in Table 2. It is observed that for ASR decoding, the rescoring method is still better with the RNN compared to the TDNN, although the TDNN is better than the n-grams. The reason could be that the sentences are not very long and hence the effective long history which is a property of the TDNN is not being exploited much. Nevertheless, the TDNN can still be used for ASR second pass decoding since it is faster to train compared to the RNN.

Table 2. **WER using Second Pass Decoding on the Babel Set of Tagalog and Swahili**

	WER (MBR)		WER (Chain)	
Number of layers↓	Tagalog	Swahili	Tagalog	Swahili
n-gram	47.9	44.7	45.5	41.4
RNN	46.5	44.0	44.4	41.0
TDNN	47.0	44.3	45.4	41.1

Table 3. **WER with varying number of layers of TDNN on the Babel Set of Tagalog and Swahili**

	WER (MBR)	
Number of layers↓	Tagalog	Swahili
2-layer	47.0	44.3
3-layer	47.2	44.4
4-layer	47.3	44.4

6.3. Word Error Rate (WER) with varying number of hidden layers of TDNN for second pass decoding

The hidden layers of the TDNN were varied to see if there was any improvement. It was observed that with increasing the number of hidden layers, the performance degrades. This is expected, since with deeper structures of TDNN, it is expected that the TDNN will cover larger effective history. However in this work of low resource ASR, the sentences are processed independently and also the length of the sentences are not that long. Hence with 2 layer TDNN which just about covers the effective history, the performance is the best as seen in Table 3.

6.4. Word Error Rate (WER) for first pass decoding

The second pass decoding shows a good potential of using TDNN as a langugage model for ASR. In this section, it is explored whether the TDNN is able to add information in terms of the probabilities for the first pass decoding.

The first step is to create n-grams using the training data. The n-gram probabilities which are count based probabilities are replaced with the probabilities estimated from the TDNN. We estimate probabilities for the bi-grams and the tri-grams from the TDNN. The unigrams are obtained by counting. It is not possible to estimate the unigram probabilities from the TDNN since there will be no context available. These n-grams which have probabilities estimated from the TDNN are used to create the finite state transducer which is then used for building the decoding graph. The results are shown in Table 4. It can be seen that for both Tagalog and Swahili the performance of the RNN and the TDNN slightly degrades compared to the n-gram. The ngram uses a smoothing technique to remove the redundant n-grams from the training data. However these n-grams may be present in the test data. These n-grams may have appeared rarely in the training data which may have caused the n-gram estimation method to smooth them out due to their low count. The available n-grams after smoothing may not change much on the WER performance even if their probabilities are estimated by the TDNN or RNN.

Another method to exploit the TDNN is to estimate the probabilities of those n-grams which have low count in the training data but may very much be present in the test data. These n-grams are extracted directly from the n-best hypothesis. The n-best hypothesis will give a good idea of the n-grams that may be present in the final decoded output. Hence instead of building all the n-grams from the training set, we use the train set to generate unigrams and the bi-grams and tri-grams are estimated from the N-best hypothesis. The bi-gram and tri-gram probabilities are estimated from the TDNN. The results are shown in Table 5. It can be seen that this method gives better WER for both TDNN and RNN compared to the count based n-gram estimation. This shows that there is potential of using TDNN as a language model for low resource ASR and hence the experiments will be continued.

The training time of the TDNN is lower than the RNN for the same amount of data and resources. It takes two hours for the TDNN while almost one day for the RNN on the same amount of training data.

Table 4. **WER using First Pass Decdoding (n-grams from training text) on the Babel Set of Tagalog and Swahili**

	WER (MBR)		WER (Chain)	
Language Model↓	**Tagalog**	**Swahili**	**Tagalog**	**Swahili**
n-gram	**47.9**	**44.7**	**45.5**	**38.2**
RNN	**48.3**	**44.9**	**46.1**	**38.9**
TDNN	**48.1**	**45.2**	**46.0**	**38.7**

Table 5. **WER using First Pass Decdoding (n-grams from n-best list) on the Babel Set of Tagalog and Swahili**

	WER (MBR)		WER (Chain)	
Language Model↓	**Tagalog**	**Swahili**	**Tagalog**	**Swahili**
n-gram	**47.9**	**44.7**	**45.5**	**38.2**
RNN	**47.4**	**44.6**	**45.1**	**38.1**
TDNN	**47.6**	**44.4**	**45.2**	**38.3**

7. CONCLUSION

The investigation of TDNN as a language model for low resource ASR has been performed in this work. The TDNN gives good perplexity compared to RNN and n-gram. However, in terms of WER, the RNN still performs better, although the TDNN performance is close to it and better than the n-gram. The TDNN is faster to train compared to the RNN. Experiments in terms of first pass and second pass decoding showed that there is potential for the TDNN to improve, since in this work, the long history property of the TDNN has not been exploited much. This is due to the shorter length of the sentences in low resource ASR. The experiments will be continued in the direction of trying to exploit the long history property of the TDNN and to incorporate it in a better way into the low resource ASR. There is potential for the TDNN to perform better as can be evident from the perplexities.

8. ACKNOWLEDGEMENT

The authors would like to thank Sibo Tong and Francois Marelli for their help and discussion.

9. REFERENCES

[1] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[2] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget, "Recurrent neural network based language modeling in meeting recognition," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[3] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Extensions of re-

current neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[4] Xunying Liu, Yongqiang Wang, Xie Chen, Mark JF Gales, and Philip C Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4908–4912.

[5] Xie Chen, Xunying Liu, Mark JF Gales, and Philip C Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5401–5405.

[6] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[7] Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, Martin Karafiát, and Sanjeev Khudanpur, "Variational approximation of long-span language models for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5532–5535.

[8] Gwénolé Lecorvé and Petr Motlicek, "Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[9] Heike Adel, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar, and Tanja Schultz, "Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[10] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang, "Phoneme recognition using time-delay neural networks," in *Readings in speech recognition*, pp. 393–404. Elsevier, 1990.

[11] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky, "Rnnlm-recurrent neural network language modeling toolkit," in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.

[12] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number EPFL-CONF-192584.