ARCHIVES OF ACOUSTICS
Vol. **40**, No. 2, pp. 191–195 (2015)

Copyright © 2015 by PAN – IPPT

DOI: 10.1515/aoa-2015-0021

Deep Belief Neural Networks and Bidirectional Long-Short Term Memory Hybrid for Speech Recognition

Łukasz BROCKI, Krzysztof MARASEK

Polish-Japanese Academy of Information Technology
Koszykowa 86, 02-008 Warszawa, Poland; e-mail: lucas@primespeech.pl

(*received September 23, 2013; accepted February 6, 2015*)

This paper describes a Deep Belief Neural Network (DBNN) and Bidirectional Long-Short Term Memory (LSTM) hybrid used as an acoustic model for Speech Recognition. It was demonstrated by many independent researchers that DBNNs exhibit superior performance to other known machine learning frameworks in terms of speech recognition accuracy. Their superiority comes from the fact that these are deep learning networks. However, a trained DBNN is simply a feed-forward network with no internal memory, unlike Recurrent Neural Networks (RNNs) which are Turing complete and do posses internal memory, thus allowing them to make use of longer context. In this paper, an experiment is performed to make a hybrid of a DBNN with an advanced bidirectional RNN used to process its output. Results show that the use of the new DBNN-BLSTM hybrid as the acoustic model for the Large Vocabulary Continuous Speech Recognition (LVCSR) increases word recognition accuracy. However, the new model has many parameters and in some cases it may suffer performance issues in real-time applications.

Keywords: deep belief neural networks, long-short term memory, bidirectional recurrent neural networks, speech recognition, large vocabulary continuous speech recognition.

1. Introduction

Deep Belief Neural Networks (DBNNs) are multilayer, densely connected, nets introduced by Hinton (Hinton *et al.*, 2006). The hidden units in DBNNs have binary values called feature detectors. The hidden layers form an associative memory. Following (Hinton *et al.*, 2013) the two most significant properties of DBNNs are: 1) a layer-by-layer procedure for learning the top-down weights that determine how the variables in one layer depend on the variables in the layer above; 2) once the network is trained, the bottom-up pass that starts with an observed data vector allows to reproduce proper hidden unit states. These properties allow the network to self-organize. DBNNs must be trained layer by layer. Eventually, a fine tuning of all weights is needed and it can be performed the same way as in Multi-Layer Perceptron networks (MLPs). A DBNN is a composition of simple learning units, which are Restricted Boltzmann machines (RBMs) (Ackley *et al.*, 1985) that contain a layer of visible units representing the input features and a layer of hidden units that learn how to represent features and capture higher-order dependencies in the data. The two layers are connected by a matrix of symmetrically weighted connections W. There are no recursive connections within a layer. Given a vector of activities v for the visible units, the hidden units (vector h) are all conditionally independent. By starting with an observed feature vector on the visible units and alternating several times between sampling from $p(h|v,W)$ and $p(v|h,W)$, it is easy to get a learning signal.

While a trained DBNN shares an identical topology to a multilayer perceptron, they utilize a much better training procedure, which begins with an unsupervised pretraining that models the hidden layers. Technically, a trained DBNN is just a feedforward network with no internal memory, unlike Recurrent Neural Networks (RNNs) which are Turing complete and posses internal memory, which allows them to make use of longer context. In this paper an experiment is performed to make a hybrid of a DBNN with an advanced RNN on top. For the purpose of the experiment a corpus of Polish TV and radio broadcasts is used.

2. DBNN-BLSTM hybrid

Although, a hybrid of DBNN-BLSTM is already described in the literature (Wollmer *et al.*, 2009; Graves *et al.*, 2013), this paper presents new implementation and is the first attempt to use this technology for the Polish language. Moreover, unlike other papers, this work describes the use of DBNN-BLSTM hybrid in a complete LVCSR system.

Long-Short Term Memory (LSTM) neural nets are a special type of Recurrent Neural Networks (RNN), which can be trained using the gradient descent method. The characteristic topology of these networks guarantees that the backward propagated error remains at a constant level. Depends on task such networks can learn dependencies 1000 time steps apart (Hochreiter *et al.*, 1995). This is a feature difficult to obtain by many standard RNNs. An LSTM network consists of the so-called memory blocks. Each block contains three gates which control a special memory cell used for storing information. The gates are actually simple perceptrons with sigmoidal activation functions. The values returned by these functions are limited to the ⟨0, 1⟩ range. The input of the memory cell is multiplied by the output of the input gate. Therefore, if the input gate returns a value close to zero this will stop the signal trying to reach the memory cell. If the value of the input gate is close to one, the signal will reach the memory cell almost unchanged. The output gate works almost the same way. The only difference is that it interacts with the output of the network. There is one more gate inside the memory block used for resetting of the memory cell. If the value of the forget gate is close to zero the value saved in the memory cell will be erased, but if it is close to one, it will remain unchanged. A very good analogy to the memory block is an electronic chip which performs read, store and reset memory operations. The memory block of the LSTM is actually a differentiable version of such a chip. Figure 1 shows a single LSTM blocks.

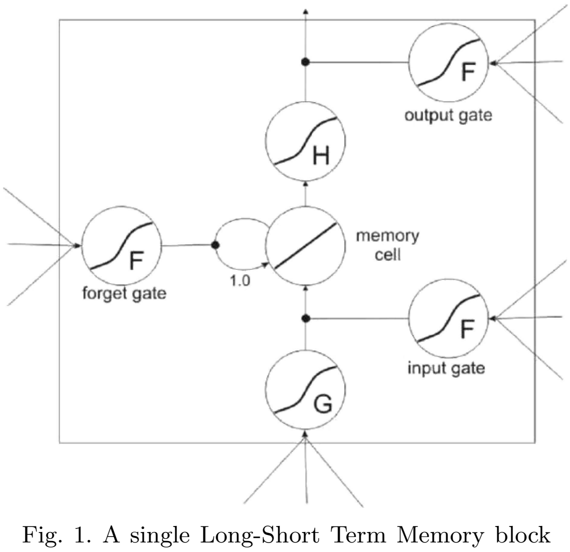

Fig. 1. A single Long-Short Term Memory block and it's circuitry.

Bidirectional Long-Short Term Memory (BLSTM) is a network consisting of two unidirectional LSTM models. The input features are presented forwards and backwards to two separate recurrent networks, both of which are connected to the same output layer. This allows the model to make use not only of past context but also future context of the presented sequence. The Bidirectional Recurrent Neural Networks (RNNs) were first introduced in (Schuster *et al.*, 1997) and they were further extended to BLSTMs. Graves *et al.* (2005) show that BLSTMs are superior to most other neural network frameworks available by 2005 in terms of acoustic modeling. In 2006 the Connectionist Temporal Classification (CTC) training algorithm was introduced and allowed to further improve phoneme recognition accuracy (Graves *et al.*, 2006). However, in 2009 Hinton *et al.* shows that DBNNs are superior even to BLSTM-CTC (Mohamed *et al.*, 2009). The superiority of DBNNs comes from the fact that when networks with many hidden layers are applied to structured data like speech, backpropagation algorithm works much better if the weights in the hidden layers are first initialized by learning the model of the structure in the input data.

The DBNN-BLSTM hybrid is trained like a regular DBNN network, however the final weight tuning is performed using the CTC algorithm which is proven to generate better results than ordinary framewise classification (Graves *et al.*, 2006). A DBNN-BLSTM hybrid consists of a single DBNN which is trained iteratively layer-by-layer. However, on top there are two LSTM networks: one processing in forward direction, the other in backward direction. Input to both LSTM networks is simply the output from the last DBNN layer. Finally, both LSTM models are merged together with the same output layer using a Softmax activation function (Bishop *et al.*, 1995). The main idea behind this hybrid is that a single DBNN network will act as a feature detector much better than the LSTM itself and the BLSTM network on top will act better then standard vector of simple perceptrons used typically in DBNNs. In order to find out which topology is better one should compare directly DBNN with DBNN-BLSTM hybrid. However, this was not possible as our ASR system uses CTC training algorithm, which was designed with RNNs in mind. Therefore a comparison between standard BLSTM and DBNN-BLSTM hybrid (both trained with CTC) was made.

3. LVCSR architecture

The system used in the paper is an evolution of the authors' system used in previous projects (Korzinek *et al.*, 2011). The acoustic layer is a BLSTM trained with the CTC algorithm. The feature front-end is the standard 12 MFCCs and energy with delta and accelertation coefficients (Young, 2000). The speech is analyzed using a 25-ms Hamming window with a 10-ms fixed frame step. The system uses a modified

Viterbi-style decoder extended to utilize the acoustic model CTC outputs properly. On the TIMIT corpus it achieved 74% phoneme recognition rate, which is comparable to other systems (Graves, 2006). The system was previously designed to work with a grammar-based language layer used to successfully recognize domain-constrained tasks with up to several thousands of concepts per grammar state. It achieved word recognition rates in excess of 90% which was also true when the system was used in noisy telephony environments. A preprocessing subsystem was independently trained to perform preliminary sound level normalization and speech detection. The normalization was performed in batch to achieve unit variance and zero mean on all files individually. The speech detection was then trained in a similar fashion to the acoustic model to recognize speech, non-speech and silence events. The training was performed on a small development corpus and achieved >90% accuracy on the test set. For the purposes of the current system, a language model was developed to aid the process of LVCSR. Since Polish language is highly inflected, a straightforward n-gram approach does not work too well. The inflection causes the vocabulary size to grow several times compared to languages like English. Larger vocabularies usually require more training data and they considerably reduce the speed and accuracy of the decoding process. On the other hand, the different inflected forms of a particular lemma play a consistent role in the sentence structure. That is why a combination of three language models responsible for words, word lemmas and grammar classes was used. The models were linearly interpolated and their interpolation weights were determined using a machine learning approach minimizing the perplexity on a validation set.

For language modeling, the transcriptions of 100 hours of radio and TV broadcasts were primarily used, but even the largest acoustic corpora make for weak language corpora, because the amount of text data is limited. Therefore this was combined with the usual up-to-date data available online (newspapers, blogs, encyclopedias). The final model was built from 3 different sub-models: a model of grammar features, word lemmas and words themselves combined together using linear interpolation. An Evolution Strategy that minimized the perplexity of the final model on the development set was used to find the optimal weights of this interpolation process. The weights of the different corpora were also optimized in this process. The language models were initially trained on a manually prepared language corpus (Korzinek, 2011) where they achieved a perplexity of 376 with a lexicon of around 40K words. They were then used to annotate and disambiguate the texts of the core training and test sets. After retraining the models with the new data the perplexity on the core test set was 246.

4. Experimental setup

To train the acoustic models, a database of TV and radio broadcasts was recorded using a DTV/radio tuner hardware and downloaded from online streams. A collection of around 120 hours of various shows was saved in 16 kHz 16-bit uncompressed audio format. The database was then manually transcribed as obtaining accurate transcriptions turned out to be impossible. To save time, all incomplete and inaudible words (as judged by the transcribers) were also removed. This greatly helped in acquiring a completely correct data set useful for training of clean acoustic models.

Word recognition experiments were performed on the mentioned corpus. The data set was split into 90% for training and development and 10% for testing. However acoustic models (BLSTM and DBNN-BLSTM hybrid) were trained only on ca. 12 hours of speech. This was due to speed issues that are described in sections 6 – Discussion. The MFCC speech features were normalized so that, they had zero mean and unit variance. We used 36 target class labels (36 phonemes). The DBNN had 7 layers with 1024 units in each layer. Mohamed (2009) empirically proved that adding more hidden layers gives better performance, although the gain diminishes as the number of layers increases. Using more hidden units per layer also improves performance. Because of computational and time limitations only one DBNN topology was tested in this research. DBNN was trained for 75 epochs with a learning rate of 0.005 and a momentum rate of 0.9. The RBM units were binary. A final tuning was performed after adding a BLSTM layer with 200 blocks (100 forward and 100 backward). A learning rate was empirically set to of 10^{-7} and momentum was set to 0.9. BLSTM initial synaptic weights were uniformly randomized between -0.1 and 0.1. These settings were used for both BLSTMs (standalone and DBNN-BLSTM hybrid). However, a standalone BLSTM had only 39 inputs (speech features) and circa 113.000 synaptic connections. A BLSTM used in the hybrid had 1024 inputs which increased the number of weights to over 900.000 slowing down training significantly.

Trained acoustic models were incorporated to already working LVCSR system (Korzinek, 2011). A standard system based on GMM models was also trained in HTK toolkit (Young, 2000) and tested in the Julius (Lee, 2001) decoder using a language model created by IRSTLM toolkit (Federico, 2008). One must underline that GMMs were trained using full corpus (120 hours of speech). However, this system performed considerably worse compared to our own, which was trained only on 12 hours of speech. It seems that worse result may be due to the sensitivity of GMMs to challenging acoustic data and less advanced language model.

Table 1. Results of experiments.

Decoder	Accoustic model	Acoustic corpus size (hours)	Lexicon size	Language model perplexity	Word accuracy
Viterbi-like	BLSTM	120	40K	246	65%
Viterbi-like	DBNN-BLSTM	12	40K	246	61%
Viterbi-like	BLSTM	12	40K	246	56%
Julius+IRSTLM	GMM	120	60K	276	47%
Julius+IRSTLM	GMM	120	30K	289	44%

5. Results

Experiment results are summarized in the Table 1. The highest word accuracy was achieved by BLSTM acoustic model trained on the full corpus (120 hours). However, if BLSTM was trained only on 12 hours of speech it's performance dropped drastically and was worse then DBNN-BLSTM hybrid. It is very indicative that if one trained DBNN-BLSTM hybrid on the full corpus it would have achieved best result. However, this was not possible due to the amount of time that would be needed to train such a big model. It must be also highlighted that all systems in Table 1 apart from DBNN-BLSTM hybrid work close to or faster than real time.

The size of the vocabulary was limited to 40K words because of the design issues and memory requirements of the current system implementation. Most common recognition errors were related to Out-Of-Vocabulary words in a limited vocabulary language model. Another problem would often arise when speakers suddenly changed, which is very common in news and talk shows. It seems that these rapid changes would interfere with the acoustic model contexts. Finally, background noise and many people speaking at the same time would cause greatest errors. Even though the data set was cleaned to remove all the non-speech and inaudible speech, it was decided to leave background noise in the recordings as long as the person transcribing the data could understand the spoken word without too much effort.

6. Discussion

Training DBNNs for speech recognition is computationally very expensive. Therefore many researchers accelerate processing by incorporating graphics cards (Dahl, 2011). Mohamed *et al.* (2009) uses GPUs in a NVIDIA Tesla S1070 system, together with the CUDAMAT library. They report that single GPU learns at 20 times faster than a single 2.66 GHz Xeon core. This is a significant improvement that allowed the researchers to make more experiments with different settings and achieve better results. Unfortunately, in this work no GPUs were used as BLSTM trained with the CTC algorithm is quite complex to be easily implemented for GPUs. GPUs are great if they are to multiply or add large matrices and that is exactly what is needed for DBNNs (as they are similar to MLPs). On the other hand BLSTM are bidirectional complicated networks built from blocks with much more advanced recursive circuitry then standard feed-forward networks like DBNNs. Moreover, as BLSTMs are RNNs they are trained with a Backprogragation Through Time algorithm (Werbos, 1987) which "unfolds" the networks states through all time frames and because of that it needs significant amount of memory. The last aspect that makes GPUs far from perfect solution in case of DBNN-BLSTM hybrid is the CTC learning algorithm, which uses dynamic programming, similar to the forward-backward algorithm for HMMs (Rabiner, 1989). The key idea of CTC algorithm is that the sum over paths corresponding to a labeling can be broken down into an iterative sum over paths corresponding to prefixes of that labeling. The iterations can then be computed with recursive forward and backward variables. Considering all these problems the authors decided to implement their programs in standard C without GPU acceleration.

A properly trained DBNN-BLSTM hybrid is superior to BLSTM in terms of speech recognition accuracy. However, when it comes to using a trained model it is obvious that it runs orders of magnitude slower then a compact BLSTM (not to mention GMM-HMM tandem which is even faster). A standalone BLSTM used in this experiments has 39 input features, 100 blocks (in each direction) and 38 output neurons (36 phonemes + 1 so-called "blank" unit (Graves, 2006)). The whole model has around 113.000 weights and can be run in real time (100 passes per second) on every standard PC. The DBNN-BLSTM hybrid is a completely another case. The DBNN net alone (without the BLSTM on top) has 7 layers consisting of 1024 units each. Consecutive layers are densely connected with each other which gives 1.048.576 weights for each layer. The whole DBNN net alone has almost 6.5 million weights. Adding BLSTM on top makes things even worse. The BLSTM in the hybrid has the same number of blocks as standalone BLSTM, however it has 1024 input features apart from 39. Therefore the top layer BLSTM has 900.000 weights in comparison to 113.000 in standalone BLSTM. The whole DBNN-BLSTM model has much over 7 million synaptic connections which makes this model over 60 times larger then standalone BLSTM. Running such model in real time on current PCs without any advanced accelerations is truly impossible. Training the DBNN-BLSTM hybrid is also a very computationally demanding process and that is the reason why in these experiments only 12 hours of audio were used for training.

7. Conclusions

This paper presents recent work on extending the DBNN to a DBNN-BLSTM hybrid. A framewise-phoneme-based training is further extended with Connectionnist Temporal Classification algorithm. The DBNN-BLSTM hybrid is a powerful topology that is superiors in terms of speech recognition accuracy when compared to standalone BLSTM and GMM. We have shown that DBNN-BLSTM hybrid can be implemented for LVCSR. Although our experiments show that DBNN-BLSTM provide improvements in recognition accuracy, training such models is computationally much more expensive compared to BLSTM. Moreover, DBNN-BLSTM hybrid has 60 times more synaptic connections then BLSTM and therefore it is not possible to run it in real time which might generate issues in real-world applications. The authors believe that this research is only the first step towards a more accurate and speed efficient acoustic models for LVCSR.

Acknowledgment

This work was financially supported by the European Community from the European Social Fund within the INTERKADRA project UDA-POKL-04.01.01-00-014/10-00, NCN N516519439 grant and Eu-Bridge project (FP7 grant agreement no. 287658).

References

1. Ackley D., Hinton G., Sejnowski T. (1985), *A Learning Algorithm for Boltzmann Machines*, Cognitive Science, **9**, 1, 147–169.
2. Bishop C.M. (1995), *Neural networks for pattern recognition*, Oxford University Press, ISBN 0-19-853864-2.
3. Brocki Ł., Korzinek D., Marasek K. (2006), *Recognizing Connected Digit Strings Using Neural Networks*, TSD 2006, Brno, Czech Republic.
4. Dahl G.E., Yu D., Deng L., Acero A. (2011), *Large vocabulary continuous speech recognition with context-dependent DBN-HMMS*, ICASSP 2011, 4688–4691.
5. Federico M., Bertoldi N., Cettolo M. (2008), *IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models*, Proceedings of Interspeech, Brisbane, Australia.
6. Graves A., Fernandez S., Schmidhuber J. (2005), *Bidirectional LSTM networks for improved phoneme classification and recognition*, ICANN 2005, Warsaw, Poland, pp. 799–804.
7. Graves A., Rahman A., Hinton G.E. (2013), *Speech Recognition with Deep Recurrent Neural Networks*, ICASSP 2013, Vancouver, Canada.
8. Graves A., Fernández S., Gomez F. Schmidhuber J. (2006), *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks*, ICML 2006, Pittsburgh, USA, pp. 369–376.
9. Graves A., Eck D., Beringer N., Schmidthuber J. (2004), *Biologically Plausible Speech Recognition with LSTM Neural Nets*, Bio-ADIT 2004, Lausanne, Switzerland, pp. 175–184.
10. Hinton G.E., Osindero S., Teh Y. (2006), *A fast learning algorithm for deep belief nets*, Neural Computation, 18, pp 1527-1554
11. Hochreiter S., Schmidhuber J. (1995), *Long Short-Term Memory*, Neural Computation, **9**, 8, 1735–1780.
12. Korzinek D., Marasek K., Brocki Ł. (2011), *Automatic Transcription of Polish Radio and Television Broadcast Audio*, Intelligent Tools for Building a Scientific Information Platform, Springer, **467**, 489–497.
13. Lee A., Kawahara T., Shikano K. (2001), *Julius an open source real-time large vocabulary recognition engine*, [in:] Proc. European Conference on Speech Communication and Technology (EUROSPEECH), pp. 1691–1694.
14. Mohamed A., Dahl G.E., Hinton G.E. (2009), *Deep belief networks for phone recognition*, [in:] NIPS Workshop on Deep Learning for Speech Recognition and Related Applications.
15. Rabiner L.R. (1989), *A tutorial on hidden markov models and selected applications in speech recognition*, Proc. IEEE, pp. 257–286.
16. Schuster M., Paliwal K.K. (1997), *Bidirectional recurrent neural networks*, IEEE Transactions on Signal Processing, **45**, 2673–2681, November 1997.
17. Stolcke A. (2002), *SRILM – An Extensible Language Modeling Toolkit*, Speech Technology and Research Laboratory SRI International, Menlo Park, USA.
18. Werbos P.J. (1987), *Backpropagation through time: what it does and how to do it*, Proc. IEEE, **78**, 10, 1550–1560.
19. Wollmer M., Eyben F., Graves A., Schuller B., Rigoll G. (2009), *A Tandem BLSTM-DBN architecture for keyword spotting with enhanced context modeling*, NOLISP 2009, Vic, Spain.
20. Young S. (2000), *The HTK Book*, Cambridge University Press.
21. www.scholarpedia.org/article/Deep_belief_networks