Deep Neural Network based Text-Dependent Speaker Recognition: Preliminary Results

Gautam Bhattacharya, Jahangir Alam, Themos Stafylakis, Patrick Kenny

Computer Research Institute of Montreal
Montreal, Canada

Abstract

Recently there has significant research interest in using neural networks as feature extractors for text-dependent speaker verification. These types of systems have been shown to perform very well when a large amount of speaker data is available for training. In this work we are interested in testing the efficacy of these methods when only a small amount of training data is available. Google recently introduced an approach that makes use of Recurrent Neural Networks (RNNs) to generate utterance-level or global features for text-dependent speaker verification. This is in contrast to the more established approach of training a Deep Neural Network (DNN) to discriminate between speakers at the frame-level. In this work we explore the DNN (feed forward) and RNN speaker verification paradigms. In the RNN case we propose improvements to the basic model with respect to the small training set available to us. Our experiments show that while both DNNs and RNNs are able to learn the training data, the set used in this study is not large or diverse enough to allow the them to generalize to new speakers. While the DNN models outperform the RNN, both models perform poorly compared to a GMM-UBM system. Nonetheless, we believe this work serves as motivation for the further development of neural network based speaker verification approaches using global features.

1. Introduction

Text-dependent speaker verification is a two-step process. The system must verify a persons identity while insuring that the correct pass-phrase was spoken. The latter implicit implies that the system has knowledge of the utterance's phonetic content. This knowledge can then be used to constrain the speaker modelling process [1]. This leads to two major differences between text-dependent and text-independent speaker verification systems. Firstly, text-dependent system can be trained with far less background data. Secondly, text-dependent system are able to achieve robust verification results with utterances of short-duration.

Deep Neural Networks (DNNs) are now well established as a means to learn useful representations or abstractions of real world data. In text-dependent speaker verification DNNs are not used for classification directly, but rather as *feature extractors*. The approach is two fold. First a (often deep) neural network is trained to discriminate between speakers from a training set (this is a classification task). After the network is trained, it is used to extract features for the enrolment and test speakers. This is done by freezing the parameters of the network and removing the output layer. Typically, the enrolment and test features are scored using a simple classifier like cosine distance. We note that the background / training set used in this work is too small to train a classifier like Probabilistic Linear Discriminant Analysis (PLDA). The performance of such systems depends quite heavily on the amount of background data available. Indeed, neural network based speaker verification approaches have been able to outperform an i-vector/PLDA system on very large datasets.

In this work we examine the use of neural networks for text-dependent speaker verification with a small background set. Specifically, the background set consists of 98 speakers (both male and female), with 8-15 recordings per speaker. Neural networks are notoriously difficult to train on small datasets. The most common strategy to overcome this shortcoming is to use generative pre-training, or regularization techniques like dropout [2, 3].

Our findings indicate that on a dataset of this size a neural network is able to memorize the training set, and is able to achieve excellent results on closed-set speaker identification (based on validation set results). Crucially however, the models are unable to generalize well to new speakers. This fact is reflected in the poor performance of the DNN models compared to a GMM-UMB system.

We focus on the use of Recurrent Neural Networks (RNNs) for building feature extractors for speaker verification. Contrary to most of the literature on DNN based text-dependent speaker verification, we explore and extend a new paradigm for text-dependent speaker recognition introduced in [4].
We also experiment with the d-vector paradigm [5], which makes use of a DNN for feature extraction. The primary difference between the two approaches is that the d-vector approach classifies each frame of a recording during training, whereas the RNN approach classifies each sequence or recording. In principle a DNN can be used to make classifications at utterance level rather than frame level. The authors of [4] showed that this approach works better, and a RNN outperforms the DNN. Interestingly, our results are contrary to these findings. Indeed the d-vector approach is able to produce better results than the RNN approach. We believe that the main reason for this is the fact that the d-vector approach operates at the frame level. While operating at the utterance level seems like the most natural thing to do, working at the frame level gives us many more data-points. The improvement in speaker verification performance over the RNN is small, however this gap is more impressive in the case of closed-set identification.

The rest of the paper is organized as follows. In section 2 we present a survey of deep neural network approaches for text-dependent speaker verification. Section 3 gives an overview of Recurrent Neural Networks (RNNs). Section 4 builds on the the further elaborates on the idea of utterance-level speaker representation, and proposes some enhancements. In section 5 we present the details regarding the dataset, model training, as well as speaker verification results. In Section 6 we summarize our findings and directions of future work.

2. DNNs for Speaker Verification

In this section we present a survey of DNN based approaches for speaker verification. DNNs have successfully been integrated into both text-independent and text-dependent verification paradigms. Two types of DNNs have been used in speaker verification - phonetic discriminant DNNs and speaker discriminant DNNs. All the techniques detailed in this section make use of spectral features as input to the neural network.

2.1. Phonetic Discriminant DNN

A phonetic discriminant DNN refers to a neural network that classifies each frame of speech as a specific phoneme (or triphone). That is, a DNN trained for speech recognition.

2.1.1. DNNs for collecting Sufficient Statistics

The most successful application of DNNs to speaker verification involves the use of a DNN speech recognizer to replace the GMM-UBM in the i-vector approach. The UBM is used to collect sufficient statistics in i-vector approach. The basic idea is to replace the frame alignment posteriors generated by the GMM with the senone posteriors produced by the DNN [6, 7]. To our knowledge this approach has only been applied in a text-independent setting.

2.1.2. Phonetic Bottleneck Features

Another approach that makes use of a phonetic discriminant DNN for speaker verification is the so-called bottleneck or tandem features approach [8]. A DNN is trained in supervised mode using the triphone state labels as targets. Once the network is trained, deep features can be extracted for every speech frame of a recording. The dimension of the deep feature is usually kept the same as the spectral feature by means of a bottleneck layer. These features in turn are used to train a backend classifier like a GMM-UBM or PLDA. Tandem feature are formed by combining the deep feature and the spectral feature corresponding to a given speech frame.

Tandem features have been successfully applied to both text-independent and text-dependent speaker verification [9, 10]. In the text-dependent case, the approach was tested on the multiple pass-phrase task. To our knowledge this works represents the smallest dataset used for training neural net models.

2.2. Speaker Discriminant DNN

A speaker discriminant DNN is a neural network trained to discriminate between speakers. This type of network would represent the most natural configuration for speaker recognition / verification. The predominant approach involves training a supervised DNN to discriminate between speakers at the frame level. The main criticism of this technique is that speaker discrimination is done based on a time-scale at which phonetic variability is the dominant type of variability.

2.2.1. d-vectors

In this approach a DNN is trained to discriminate between speakers at the frame-level. Each speech frame is stacked along with a fixed number of context frames and is fed as input to the network. The DNN then needs to classify each frame as belonging to 1-of-N speakers. Where N is the number of background speakers. This can be seen as the speaker recognition (not verification) equivalent of a speech recognition DNN. The approach was studied on the common passphrase task [5].

Once training is complete, the output of the last hidden layer of the network is used to produce an utterance-level feature for a speaker. That is, each frame of an utterance is forward propagated through the network, and the hidden activations of all the frames are averaged to form an utterance-level feature called a d-vector. The enrolment speaker models are formed by averaging the d-vectors corresponding to the enrolment recordings. Recently, a similar approach has been studied with convolutional networks [11], however a much larger background set is used here to tackle the same task.

2.2.2. Multi-task Learning

The authors of [12] approach the multiple pass-phrase task using multi-task learning. The idea is to train a DNN to make classifications for both the speaker and phrase identities. This is achieved by minimizing a composite loss function consisting of a sum of two cross-entropy losses - one related to the speaker label and the other to the phrase label. Once again, after training is complete the DNN is used as a feature extractor to generate features for a backend classifier. One of the interesting findings in this work is that the speaker verification performance of deep features is much worse when using the cosine distance metric as compared to more sophisticated classifiers like a GMM-UBM or PLDA. The DNN features can also be combined with the spectral features to produce tandem features.

2.2.3. Global Features

Google recently introduced a novel approach for text-dependent speaker verification that makes use of global features [4]. This work also tackles the common passphrase problem. This method addresses the main drawback of the d-vector approach, namely, frame-level speaker classification. This is done by producing an utterance-level feature for classification, rather than classify each speech frame. That is, there is a single label associated with each utterance or sequence, and not one for every frame in an utterance.

In the case of a DNN this is achieved by averaging all the hidden activations of a recording before passing the averaged, utterance-level feature to the softmax layer of the network. This work also introduces the use of recurrent neural networks to produce utterance-level features. In the case of a RNN, the trick of averaging the hidden activations is not needed as the the hidden activation corresponding to the last time-step of the recording is effectively a summary or a global feature of the entire recording.

The other major contribution of this work is the development of a novel end-to-end system. This is achieved by using the a DNN or RNN to first encode both enrolment and test recordings. Utterance-level features from both recordings are then scored using cosine distance before being passed to a logistic regression layer. Finally a binary decision is made if the two recordings should be accepted or rejected. The whole network is trained by optimizing the end to end loss.

While both the utterance-level speaker softmax and end-to-end approaches achieve excellent results, the dataset used in this work is quite large and hence of potentially less practical use. Nonetheless, we believe that the use of RNNs to process speech data and the concept of utterance-level classification for training DNN feature extractors is a promising direction of research, and choose to explore it further in this work.

3. Recurrent Neural Networks

Recurrent neural networks (RNNs) can be viewed as a generalization of of feed-forward nets to sequences of arbitrary duration. A recurrent network can be converted into a feed-forward network by unrolling its computational graph in time [13]. RNNs have been successfully applied to several sequential problems such as speech and handwriting recognition, machine translation and language modelling [14, 15, 16].

Another feature that distinguishes a RNN from a DNN is the recurrent connection (and associated weight matrix), which makes these networks especially well suited to processing time-series data. This means that while processing a given time-step of a sequence, the network has a 'memory' of all the previous time-steps. In terms of the probabilistic model, this recurrent connection allows the model to condition its prediction at the current time-step on all previous time-steps. For a detailed treatment of RNNs we refer the reader to [17, 18].

Given a sequence $X = \{x_1, x_2, ..., x_T\}$The forward hidden activation of a RNN is computed sequentially for every time-step as:

$$h_{forward} = f(W_{ih_f} x_t + W_{hh_f} h_{t-1} + b_{h_f}) \qquad (1)$$

Similarly a backward activation can be computed by considering the sequence in reverse:

$$h_{backward} = f(W_{ih_b} x_{\hat{t}} + W_{hh_b} h_{\hat{t}-1} + b_{h_b}) \qquad (2)$$

A bi-directional RNN processes a sequence in both the forward and backward directions [19]. It essentially consists of two RNNs, one for the forward and the other for the backward direction. The hidden activations of the forward and backward RNNs are then combined via concatenation. For a given time-step, an encoding vector is obtained by concatenating the forward and backward activations corresponding to that time-step.

$$h = [h_{forward}\,; h_{backward}] \qquad (3)$$

This concatenated hidden activation is passed to an output layer which computes:

$$\hat{y} = W_{ho}.h + b_o \qquad (4)$$

$$O = g(\hat{y}) \qquad (5)$$

The choice of non-linear activation function f in equations (1) and (2) is an important factor while working with RNNs. A simple RNN with sigmoid activation functions is known to suffer from the vanishing gradient problem [20]. A popular choice to avoid this difficulty is to use RNNs with Long-Short Term Memory (LSTM) cells [21]. LSTMs were specifically designed to avoid the vanishing gradient problem. Recently, Gated Recurrent Units (GRU) were proposed to also avoid the vanishing gradient problem [22].

4. Utterance-Level Features

The key idea behind using a RNN to produce utterance-level features is based on their ability to generate a sequence summaries. Given a sequence $X = \{x_1, x_2, ..., x_T\}$, we can forward propagate the sequence through an RNN to produce a recurrent embeddings $R = \{r_1, r_2, ..., r_T\}$. The recurrent embedding corresponding to the last time-step of the sequence, r_T, can be considered as the summary of the entire sequence. We refer to this recurrent embedding as the *summary vector*. This summary vector is then used by the RNN to predict the speaker's identity by means of a softmax layer, which produces a probability distribution over the speakers in the training set.

We build on this approach by introducing bi-directional recurrence into the model, as well as some other enhancements that are motivated by the data-sparse nature of our problem.

4.1. Last-Step Speaker Representation

This model is a version of the approach in [4], the only difference being the inclusion of bi-directional recurrence in the model. Here a bi-directional RNN is used to extract an utterance-level feature from a recording and then use this feature to discriminate between speakers in a background set. Unlike a vanilla RNN, only the last hidden activation corresponding to the last time-step is connected to the softmax layer.The model processes (forward-propagates) each utterance and learns a corresponding summary vector.

$$h_{forward}(t) = f(W_{ih_f} x_t + W_{hh_f} h_{t-1} + b_{h_f}) \qquad (6)$$

$$h_{backward}(t) = f(W_{ih_b} x_t + W_{hh_b} h_{t-1} + b_{h_b}) \qquad (7)$$

For a sequence $X = \{x_1, x_2, ..., x_T\}$, the concatenation hidden activations $h_{summary} = [h_{forward}(T); h_{backward}(1)]$ corresponds to the summary vector. During *training*, the summary vector is subjected to a linear transformation and is fed to a softmax activation function. The softmax function represents a distribution over speakers in the training set.

$$\hat{y} = W_{ho}.h_{summary} + b_o \qquad (8)$$

$$O = softmax(\hat{y}) \qquad (9)$$

At *runtime*, the softmax layer is removed and the model is used to extract summary vectors for the enrolment and test utterances.

4.2. Averaged Speaker Representation

One of the drawbacks of the last-step speaker representation is that the recurrent embedding corresponding to the last time-step of the utterance needs to summarize the entire sequence. It has been shown that this approach works well for sequences between 80-100 time-steps, however performance deteriorates as the sequences get longer. In our case we are dealing with sequences ranging from 150-700 frames and hence we can expect the speaker-softmax approach to struggle given this data.

A simple approach to alleviate this problem is to average all the recurrent embeddings, and feed the resulting feature to the softmax layer. This also insures that all of the data is used in order to make classification decisions, which is helpful for speaker verification performance under sparse-data conditions.

4.3. Learned Speaker Representation

Rather that simply average the hidden activation of recording to produce an utterance level feature, we propose to learn the best combination of these activations. The basic idea is to use a small feedforward neural network to learn the best way to combine recurrent embeddings (weighted sum) of an utterance in order to classify the utterance correctly. This approach is motivated by the notion of an attention model, originally introduced for machine translation [23]. In the context of our problem, the network serves as a 'combination' model.

As before, the first step involves forward propagating the input sequence through the RNN:

$$h_{forward}(t) = f(W_{ih_f} x_t + W_{hh_f} h_{t-1} + b_{h_f}) \tag{10}$$

These hidden activations are then processed by a small neural network (combination model) which assigns a scalar score to each hidden activation/recurrent embedding. The combination model is parameterized by a single-layer feed-forward neural network. The individual scores are then normalized using the softmax function. This normalization also provides a probabilistic interpretation of the scores assigned by the attention model. The output of the softmax function can then be interpreted as a sequence of weights, one for each time-step. These weights can be used to produce a weighted sum of the RNN encoding.

$$e_j = a(h_{forward}(j)) \tag{11}$$

$$c = \frac{e^{e_j}}{\sum_j e^{a_j}} \tag{12}$$

$$\hat{f} = \sum_{i=1}^{T} h(i).c_i \tag{13}$$

Where e_j represents the combination model, c is the softmax distribution over recurrent embeddings and $h(i)$ represents a given hidden activation. Finally, the weighted-sum vector $\hat{f}$ is linearly transformed and passed to the softmax function.

$$\hat{y} = W_{ho}.\hat{f} + b_o \tag{14}$$

$$O = softmax(\hat{y}) \tag{15}$$

Note that this second softmax is a distribution over the speakers in the training set.

5. Experiments and Results

In this section present details regarding the network architectures, network training and speaker verification performance. All the RNN models studied in this work were built using Theano and the Lasagne deep learning package [24, 25].

5.1. Datasets

In this work we are interested in evaluating the performance of recurrent neural networks on a small dataset. For the purposes of this preliminary study, we use a proprietary dataset consisting of a single pass-phrase. The development set consists of multiple sessions from 98 speakers, totalling 1547 recordings. For testing there are 230 unique speaker models and 1164 test utterances. Some of the enrolment speakers have been recorded under different channel conditions (multiple times). However the majority of the speaker models consist of 3 recordings each.

The spectral features used in this work consist of 20-dimensional mel-frequency cepstral coefficients (MFCC). We only use the static coefficients. We also remove silence frames from all the recordings.

For the RNN models, each recording is the training set is treated as a datapoint. The input to the DNN is a 10ms frame of speech, along with left and right context frames. We make use of a 11 frame context window around the central frame.

5.2. Network Architectures

The primary obstacle we faced during this work related to training networks on a small dataset. While most neural network architectures are known to overfit such datasets, this is especially true of recurrent networks. Nonetheless, we are interested in evaluating the performance of RNNs on this dataset as we believe it will provide useful insights for future work. We perform experiments with single-layer RNNs of different sizes, ranging from 100 to 700 hidden units.

Unlike recurrent architectures, there has been significantly more effort to improve the performance of DNNs on small training sets. Techniques like dropout regularization are now fairly commonplace in DNN training, however it is not as straightforward to apply to recurrent networks. Maxout networks [26] are another approach that has been shown to perform well on small datasets. Maxout networks differ from the standard DNNs in that hidden units at each layer are divided into non-overlapping groups. Each group generates a single activation via the max pooling operation. We experimented with DNN architectures of different depth, ranging from 3 to 7 hidden layers, and layer sizes between 200 to 700 hidden units. We made use of Rectified Linear Units (RELU) for the non-linear activation functions.

The second challenge we faced was specific to recurrent architectures. In the context of speech, RNNs operate at the level of entire recordings (variable length sequences) whereas the natural way to use DNNs is to make frame-level predictions. This is an important point as it implies that a DNN would receive many more data examples than a RNN from the same dataset. The recordings in our dataset range from 150 to 700 frames (10 ms), and as such are quite long. As stated in section 4, the key idea in the RNN speaker-verification paradigm is based on its ability to produce a fixed-size vector that summarizes the whole recording. Ideally, we would like the summary vector to memorize the entire sequence, however this becomes more challenging as sequence length increases.
One way to deal with long sequences is to increase the depth of the network [27], however this tends to make the network more prone to overfitting. We experimented with RNNs that are 2 and 3 layers deep, however these models did not perform as well as single-layer networks.

5.3. Network Training

A common feature of all the models used in this work is the softmax distribution over speakers in the training set. Consequently all models are trained by minimizing cross-entropy loss, and training is done via backpropagation. For updating model parameters, we tried several optimizers and achieved our best results using the ADAM adaptive learning rate algorithm and stochastic gradient descent with momentum for the RNN and

DNN models respectively. A learning rate of 0.001 for training all models. We also used mini-batches consisting of 32 utterances in the case of the RNN, and 256 frames (with context) for the DNN. RNNs are known to suffer from an exploding gradient problem, and consequently we clip the gradients as in [28]. For all the models, training is halted based on the principle of early-stopping [29]. In the case of networks with dropout, a dropout layer was inserted after every hidden layer of the network. A fixed dropout percentage of 0.4 was consistently used throughout our experiments.

5.4. Closed-Set Identification

While the the size of the validation set used in this work is quite small, performance on this set provides us with useful insight regarding the learning process, and this also translates to performance on speaker verification tasks. It also gives us an indication of how well the networks are able to cope with channel variability.

Interestingly, the closed-set identification performance of the RNNs and DNNs differs significantly. This is perhaps due to the larger number of data points seen by the DNN. The unidirectional RNN (basic speaker-softmax) achieved a classification accuracy of 57% on the validation set, which represents the worst performance of the networks used in this study. A bidirectional network is able to do considerably better, with a classification accuracy of 65%. The averaged speaker-softmax model does marginally better, with an accuracy of 68%. The augmentation of the attention model improves the validation accuracy considerably, achieving 82%. In the case of the DNN models, the Maxout network achieve the best classification accuracy of 98%, which was slightly better than the vanilla DNN model with RELU units that achieved 97% accuracy. Surprisingly dropout regularization did not affect performance in the case of either model (maxout and vanilla DNN), however as noted in the previous section, this hyper-parameter was not optimized.

5.5. Speaker Verification

After the RNN and DNN models are trained, we use them as feature extractors by removing the softmax layers of the networks. The RNN produces a hidden activation corresponding to every time-step in a recording. Depending on the model, an utterance level feature is produced either by averaging, a combination model or by simply taking the last hidden activation or summary vector. For DNNs, the output of the last hidden layer represents a frame-level hidden activation. We follow the standard approach of forward propagating the whole recording through the network and form an utterance-level feature by averaging the hidden activations for every time-step. In this way utterance-level features can be extracted for enrolment and test speakers. A speaker verification score is calculated using the cosine distance metric.

Table 1: *RNN Speaker Verification Results.*

Network	Architecture	EER
Unidirectional	spk-softmax	13.53
Bidirectional	spk-softmax	11.60
Bidirectional	avg-softmax	10.45
Bidirectional	attn-softmax	8.84

Table 2: *DNN Speaker Verification Results.*

Network	EER
DNN	8.48
DNN+dropout	8.25
Maxout	7.88
Maxout+dropout	7.88
Gmm-UBM	3.6

Table 1 displays the speaker verification performance of the different RNN architectures used in this study. The results correspond to our best performing models had 400 hidden units. The bidirectional speaker-softmax model clearly outperforms the unidirectional model with an ERR of 11.60 as compared to 13.53. The averaged speaker softmax representation does slightly better than the basic model with an EER of 10.45 as compared to 11.60. The attention speaker-softmax was the best performing model, achieving an EER of 8.84. This result suggests that a combination of recurrent embeddings to form an utterance-level feature represents the best approach for speaker verification, at least under data-sparse conditions.

The DNN speaker verification models show slightly better performance as compared to the RNN as reflected by table 2. The Maxout network achieves the best performance with an error rate of 7.88. We note again that dropout regularization had only a marginal effect on performance. Our best performing network consisted of 3 hidden-layers with 500 hidden units per layer. That being said, the performance of both RNN and DNN models is significantly worse that the GMM-UBM (256 component) baseline which achieves an error rate of 3.6.

6. Discussion

We also see that by asking the network to learn the best combination of hidden activations/recurrent embeddings, we are able to achieve our best speaker verification results. Moreover this approach works better than simple averaging. Consequently we are interested in integrating this model into other neural network architectures. Moreover, the approach of combining recurrent embeddings appears to produce better results than using the summary vector as a feature for text-dependent speaker verification.

7. Conclusions

In this work we explore the use of DNNs and RNNs for text-dependent speaker verification. Our main contribution is evaluating the potential of global or utterance-level features. The results of our experiments suggest that while both DNN and RNN models are able to memorize the speakers in the training set, their ability to generalize to new speakers is quite poor.

For the RNN models we experimented with three types of global features, and our experiments show that combining all the hidden activations of a recording to generate a global feature works better than using the summary vector (hidden activation corresponding to the last time-step of a sequence) as a global feature. Furthermore, using an additional neural network to learn the weights of the hidden activation leads to further improvement. The speaker verification results of these models suggest that there is room for further improvement as the RNN based systems perform poorly as compared to a GMM-

UBM system. One approach that may potentially improve performance would be to use the RNN to produce tandem features in order to train a GMM-UBM system.

The DNN based approach outperformed the RNN models on both closed-set identification as well as speaker verification. One potential reason for this is the fact that the DNN sees many more labelled datapoints during training than the RNN. In order to make the comparison fair, we could run the RNN models over sub-sequences corresponding to the input to the DNN. We hope to pursue this direction of research in future work.

8. References

[1] Anthony Larcher, Kong Aik Lee, Bin Ma, and Haizhou Li, "Text-dependent speaker verification: Classifiers, databases and rsr2015," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[2] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[4] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," *arXiv preprint arXiv:1509.08062*, 2015.

[5] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Jorge Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4052–4056.

[6] Yun Lei, Luciana Ferrer, Moray McLaren, et al., "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1695–1699.

[7] Patrick Kenny, Vishwa Gupta, Themos Stafylakis, P Ouellet, and J Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proc. Odyssey*, 2014, pp. 293–298.

[8] Tianfan Fu, Yanmin Qian, Yuan Liu, and Kai Yu, "Tandem deep features for text-dependent speaker verification.," in *INTERSPEECH*, 2014, pp. 1327–1331.

[9] Fred Richardson, Douglas Reynolds, and Najim Dehak, "A unified deep neural network for speaker and language recognition," *arXiv preprint arXiv:1504.00923*, 2015.

[10] Yuan Liu, Yanmin Qian, Nanxin Chen, Tianfan Fu, Ya Zhang, and Kai Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015.

[11] Yu-hsin Chen, Ignacio Lopez-Moreno, Tara N Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada, "Locally-connected and convolutional neural networks for small footprint speaker recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[12] Nanxin Chen, Yanmin Qian, and Kai Yu, "Multi-task learning for text-dependent speaker verification," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] Yoshua Bengio Ian Goodfellow and Aaron Courville, "Deep learning," Book in preparation for MIT Press, 2016.

[14] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[16] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *INTERSPEECH*, 2010, vol. 2, p. 3.

[17] Alex Graves, *Supervised sequence labelling*, Springer, 2012.

[18] Ilya Sutskever, *Training recurrent neural networks*, Ph.D. thesis, University of Toronto, 2013.

[19] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[20] Sepp Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[21] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[23] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[24] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a cpu and gpu math expression compiler," in *Proceedings of the Python for scientific computing conference (SciPy)*. Austin, TX, 2010, vol. 4, p. 3.

[25] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, SK Sønderby, D Nouri, D Maturana, M Thoma, E Battenberg, J Kelly, et al., "Lasagne: First release," *Zenodo: Geneva, Switzerland*, 2015.

[26] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[28] Alex Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[29] Yoshua Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, 2012.