Interspeech 2018
2-6 September 2018, Hyderabad

Memory Time Span in LSTMs for Multi-Speaker Source Separation

Jeroen Zegers, Hugo Van hamme

KU Leuven, Dept. ESAT, Belgium

jeroen.zegers@esat.kuleuven.be, hugo.vanhamme@esat.kuleuven.be

Abstract

With deep learning approaches becoming state-of-the-art in many speech (as well as non-speech) related machine learning tasks, efforts are being taken to delve into the neural networks which are often considered as a black box. In this paper it is analyzed how recurrent neural network (RNNs) cope with temporal dependencies by determining the relevant memory time span in a long short-term memory (LSTM) cell. This is done by leaking the state variable with a controlled lifetime and evaluating the task performance. This technique can be used for any task to estimate the time span the LSTM exploits in that specific scenario. The focus in this paper is on the task of separating speakers from overlapping speech. We discern two effects: A long term effect, probably due to speaker characterization and a short term effect, probably exploiting phone-size formant tracks.

Index Terms: deep learning, long short-term memory, memory leakage, memory span, multi speaker source separation

1. Introduction

Deep learning based methods are being used in many fields of speech processing. Standard feed forward neural networks cannot take into account temporal information (unless a context window is used), which seems undesirable for a speech signal. Therefor RNNs are often used which are theoretically able to memorize any input from the past. However, in practice it is found that it is difficult for the standard RNN to be trained such that it can use information from many time steps back because of phenomena known as vanishing and exploding gradients. The LSTM cell was introduced to counter these problems by using a fixed error flow with gates that control the input and output behavior of this flow [1]. LSTMs have been found to have good memory capabilities for longer time spans [2] and are currently being considered state of the art for many speech related tasks [3, 4, 5, 6, 7].

Even though it is known that LSTMs cope better with long term dependencies than standard RNNs, little research has been done to determine the memory time span for a particular task. In this paper part of the cell's memory will be leaked away on purpose to see how performance depends on lost information from the past. A related approach has been taken for the task of action detection in video [8]. A network with an LSTM component was trained and at test time the memory was cleared k time steps prior to the current time step. By evaluating the performance of the network for different k the authors roughly found the relevant time span before performance started degrading. The approach proposed in this paper differs in two ways from there approach. Firstly, the memory forgetting technique is applied during both training and testing. This prevents unexpected behavior due to mismatch (e.g. the network could behave strangely if values are changed at test time while it never happened at training time). It was indeed found that leaking all the memory only at test time gave much worse results than applying full leakage both during training and testing. Furthermore, the network might rely on information in the long term memory that it could also extract from a shorter time span. By applying the memory leakage during training as well, the network could learn to extract this information in the shorter time span. Secondly, in this paper, memory leakage is applied by limiting the output of the forget gate to a number smaller than 1 such that each time step, part of the information in the memory is lost and over time the past is forgotten. In [8] at time t the network uses a memory cell state c that was set to 0 at time $t-k$. Therefore k different cell states have to be stored that are each reset at a different time. In experiments for this paper, k could exceed a hundred, which would lead to computational problems, specifically during the back-propagation step in training. The leaking approach used in this paper preserves only a single cell state and is thus computationally more elegant, but time spans in this exponential decay approach are less precise.

The memory time span of the LSTM will be evaluated on the task of single channel multi-speaker source separation (MSSS). The original sound signals of multiple speakers have to be retrieved from a mixture with overlapping speech, without any prior knowledge of the speakers. This is thus an intraclass separation problem which forces the network to memorize some sort of speaker representation in order to consistently assign part of the mixture to the same speaker. It was even shown that explicitly presenting speaker representations at the network inputs could slightly improve MSSS accuracy [9]. It is expected that heavy memory leakage will hinder the network to retain this representation, making this tasks well suited to study long term dependencies. Common deep learning approaches for MSSS are Deep Clustering (DC) [10, 11], utterance-level Permutation Invariant Training (uPIT) [12] and Deep Attractor Networks (DANet) [13]. Experiments in this paper will be done using DC but results for other approaches are expected to be similar.

The remainder of this paper is organized as follows. In section 2 the DC approach to the MSSS task is explained, as well as the importance of speaker representations for the task. The method to leak memory will be described in section 3. Experiments will be discussed in section 4 and finally a conclusion is given in section 5.

2. Multi Speaker Source Separation

2.1. Deep Clustering

A mixture of S speakers is generally given by $\mathbf{y}[n] = \sum_{s=1}^{S} \mathbf{x}_s[n]$ where $\mathbf{x}_s[n]$ is the source signal for the sth speaker as recorded by the microphone. For the task of MSSS, signals $\hat{\mathbf{x}}_s[n]$ have to be estimated to be as close as possible to the original $\mathbf{x}_s[n]$. In the time-frequency domain, the same task can be expressed using the Short Time Fourier Transform

This work was funded by the SB PhD grant of the Research Foundation Flanders (FWO) with project number 1S66217N and the KULeuven research grant GOA/14/005 (CAMETRON).

10.21437/Interspeech.2018-2082

(STFT) of the signals. The task is then to estimate $\hat{\mathbf{X}}_s(t,f)$ from $\mathbf{Y}(t,f) = \sum_{s=1}^{S} \mathbf{X}_s(t,f)$. Usually, a mask $\hat{\mathbf{M}}_s(t,f)$ is estimated for the sth speaker such that

$$\hat{\mathbf{X}}_s(t,f) = \hat{\mathbf{M}}_s(t,f) \circ \mathbf{Y}(t,f) \tag{1}$$

for every time frame $t = 0, \ldots, T$ and every frequency $f = 0, \ldots, F$ and with $\circ$ the Hadamard product [12]. The masks are constrained by $\hat{\mathbf{M}}_s(t,f) \geq 0$ and $\sum_{s=1}^{S} \hat{\mathbf{M}}_s(t,f) = 1$ for every time-frequency bin (t,f).

In DC a D-dimensional embedding vector $\mathbf{v}_{tf}$ is found for every (t,f) via a mapping $\mathbf{v}_{tf} = f_\theta(\mathbf{Y})$. f_θ is based on a deep neural network and is chosen such that $\mathbf{v}_{tf}$ is normalized to unit length. The embedding vectors for every bin are stored as rows in a $(TF \times D)$-dimensional matrix $\mathbf{V}$. A $(TF \times S)$-dimensional target matrix $\mathbf{U}$ is defined such that $u_{tf,s} = 1$ if target speaker s has the most energy in the mixture for (t,f) and $u_{tf,s} = 0$ otherwise. A permutation independent loss function (the columns in $\mathbf{U}$ can be interchanged without changing the loss function) is then stated as

$$\begin{aligned} \mathcal{L}_\theta &= \|\mathbf{V}\mathbf{V}^T - \mathbf{U}\mathbf{U}^T\|_F^2 \\ &= \sum_{t_1,f_1,t_2,f_2} (\langle \mathbf{v}_{t_1 f_1}, \mathbf{v}_{t_2 f_2} \rangle - \langle \mathbf{u}_{t_1 f_1}, \mathbf{u}_{t_2 f_2} \rangle)^2 \end{aligned} \tag{2}$$

where $\|.\|_F^2$ is the squared Frobenius norm. Afterwards, all embedding vectors are clustered into S clusters c using K-means. The masks are then constructed as follows

$$\hat{\mathbf{M}}_{s,tf} = \begin{cases} 1, & \text{if } \mathbf{v}_{tf} \in c_s \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

Equation 1 can then be used to estimate the original source signals via the inverse STFT and overlap-add [14].

2.2. Speaker representation

A speaker representation that is often used for speaker identification tasks is the i-vector [15, 16]. If such i-vectors are explicitly presented to the input of the DC network, as was done in [9], possibly less information would have to be retained in the LSTM memory. To obtain such an i-vector, first a Universal Background Model - Gaussian Mixture Model (UBM-GMM) is trained on development data. A supervector M is derived for the utterance, using the UBM. M is then represented by an i-vector w and its projection based on the total variability space,

$$M = m + Tw, \tag{4}$$

where m is the UBM mean supervector, w is the total variability factor or i-vector and T is a low-rank matrix spanning a subspace with important variability in the mean supervector space and is trained on development data [15, 16].

3. Deep Leaky LSTM

A leaky LSTM is an LSTM that is designed to leak cell state memory. First, the architecture of the regular LSTM is explained. Secondly, the adaptation to make it a leaky LSTM is described and how the amount of leakage can be interpreted in memory time span. Finally, the memory flow in a deep network with LSTMs and how this flow can be controlled are described.

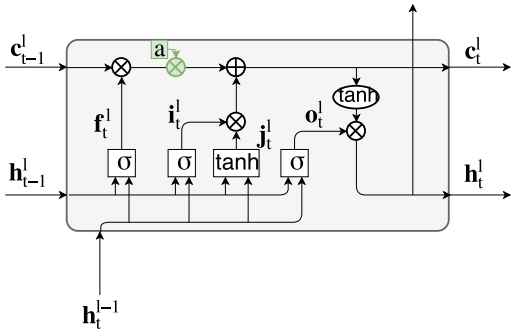

Figure 1: *Schematic of a leaky LSTM. Based on [17].*

3.1. The regular LSTM

A typical LSTM cell in a layer l of the network is defined as

$$\mathbf{f}_t^l = \sigma(\mathbf{W}_f^l \mathbf{h}_t^{l-1} + \mathbf{R}_f^l \mathbf{h}_{t-1}^l + \mathbf{b}_f) \tag{5}$$

$$\mathbf{i}_t^l = \sigma(\mathbf{W}_i^l \mathbf{h}_t^{l-1} + \mathbf{R}_i^l \mathbf{h}_{t-1}^l + \mathbf{b}_i) \tag{6}$$

$$\mathbf{o}_t^l = \sigma(\mathbf{W}_o^l \mathbf{h}_t^{l-1} + \mathbf{R}_o^l \mathbf{h}_{t-1}^l + \mathbf{b}_o) \tag{7}$$

$$\mathbf{j}_t^l = \tanh(\mathbf{W}_j^l \mathbf{h}_t^{l-1} + \mathbf{R}_j^l \mathbf{h}_{t-1}^l + \mathbf{b}_j) \tag{8}$$

$$\mathbf{c}_t^l = \mathbf{c}_{t-1}^l \odot \mathbf{f}_t^l + \mathbf{j}_t^l \odot \mathbf{i}_t^l \tag{9}$$

$$\mathbf{h}_t^l = \tanh(\mathbf{c}_t^l) \odot \mathbf{o}_t^l \tag{10}$$

with $\mathbf{h}_t^0 = \mathbf{y}_t$, where $\mathbf{y}_t$ is the network's input at time t and $\mathbf{f}_t^l$, $\mathbf{i}_t^l$ and $\mathbf{o}_t^l$ are called the forget gate, the input gate and the output gate, respectively. $\mathbf{c}_t^l$ is called the cell state, $\mathbf{h}_t^{l-1}$ is the cells input and $\mathbf{h}_t^l$ is the cell's output. $\mathbf{W}_*^l$ and $\mathbf{R}_*^l$ represent trainable weights and $\mathbf{b}_*$ trainable biases. These equations are visualized in figure 1 when the green part is ignored.

3.2. The leaky LSTM

A leaky LSTM is a regular LSTM where the forget gate is multiplied by a positive constant a smaller than 1 such that not all cell state information from the previous time steps can be retained and thus part of the memory is leaked. (9) changes to

$$\mathbf{c}_t^l = \mathbf{c}_{t-1}^l \odot \mathbf{f}_t^l * a + \mathbf{j}_t^l \odot \mathbf{i}_t^l \tag{11}$$

as is indicated in figure 1.

In (11) only the first term seems to contain temporal information since only $\mathbf{c}_{t-1}^l$ has the $t-1$ subscript. The component of the cell state containing temporal information from time $t-1$ can thus be described as

$$\bar{\mathbf{c}}_{t,t-1}^l = \mathbf{c}_{t-1}^l \odot \mathbf{f}_t^l * a \tag{12}$$

The maximal information transfer will be achieved when $\mathbf{f}_t^l = \mathbf{1}$. In this case the contribution from time $t-T$ in the current cell state is given by

$$\begin{aligned} \bar{\mathbf{c}}_{t,t-T}^l &= \mathbf{c}_{t-T}^l * a^T \\ &= \mathbf{c}_{t-T}^l * e^{T/\tau} \end{aligned} \tag{13}$$

where the last equality is written in exponential decay form with $\tau = -1/\log(a)$. τ is called the lifetime and when $T = \tau$, 69% of the initial value has been lost, which can be considered as the time span for the memory.

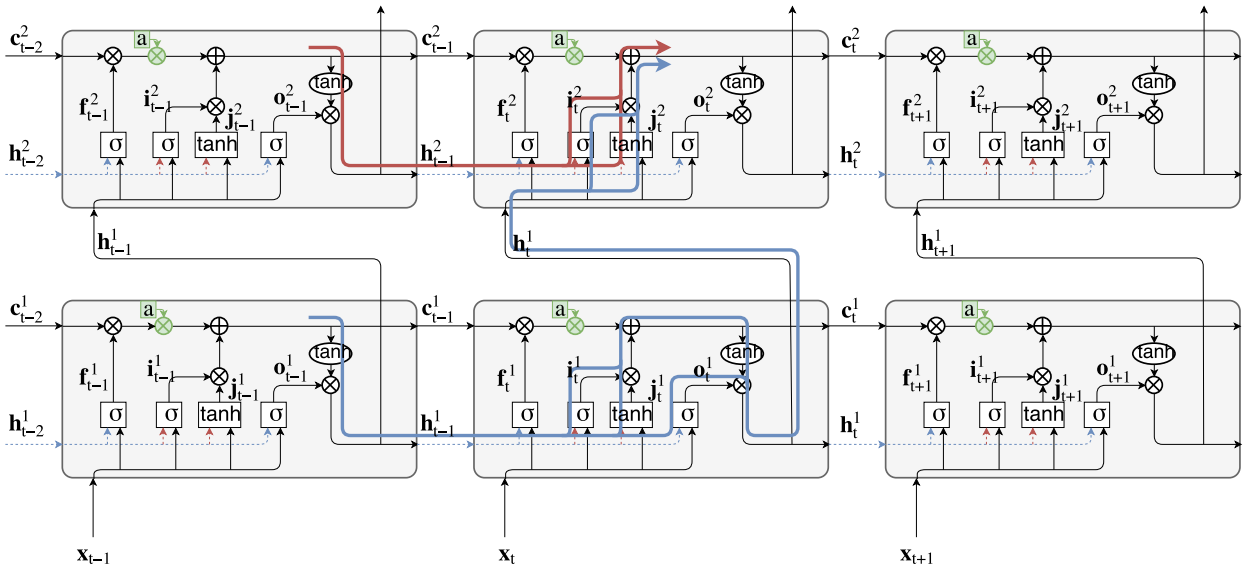

Figure 2: *Schematic of flow of temporal information in a deep leaky LSTM and how it can be interrupted by cutting connections. The red line show how recurrent information can bypass the leakage and how this can be countered by removing the red dashed connections. Similar for blue.*

3.3. Temporal information flow

It was claimed that temporal information in (11) was only present in the $\bar{\mathbf{c}}^l_{t,t-1}$ term. However, the second term in this equation has two factors that indirectly depend on previous time steps via (6) and (8). For instance, if $\mathbf{c}^l_{t-1} = \mathbf{1}$, $\mathbf{f}^l_t = \mathbf{1}$ and the network would like $\mathbf{c}^l_t = \mathbf{c}^l_{t-1} + \Delta\mathbf{c}^l_t$, it could obtains this via (11) by making $\mathbf{j}^l_t \odot \mathbf{i}^l_t = (1-a) + \Delta\mathbf{c}^l_t$ (with the restriction that $-\mathbf{1} \leqslant \mathbf{j}^l_t \odot \mathbf{i}^l_t \leqslant \mathbf{1}$). This information flow is indicated with the red arrow in figure 2. This is an unwanted flow since it bypasses the memory leakage. This flow can be stopped by cutting the red dotted connections and thus removing the temporal information in $\mathbf{j}^l_t$ and $\mathbf{i}^l_t$. Equations 6 and 8 then become

$$\mathbf{i}^l_t = \sigma(\mathbf{W}^l_i \mathbf{h}^{l-1}_t + \mathbf{b}_i) \tag{14}$$

$$\mathbf{j}^l_t = \tanh(\mathbf{W}^l_j \mathbf{h}^{l-1}_t + \mathbf{b}_j) \tag{15}$$

respectively.

However, in these equations $\mathbf{h}^{l-1}_t$ can also contain temporal information for $l \geqslant 2$ via (10) and (6,8,7,9) as is indicated by the blue arrow on figure 2. This second order flow can be stopped by also cutting the blue dotted connections. Equations 14 and 15 are retained and equations 5 and 7 become

$$\mathbf{f}^l_t = \sigma(\mathbf{W}^l_f \mathbf{h}^{l-1}_t + \mathbf{b}_f) \tag{16}$$

$$\mathbf{o}^l_t = \sigma(\mathbf{W}^l_o \mathbf{h}^{l-1}_t + \mathbf{b}_o) \tag{17}$$

respectively.

Memory leakage has only been considered on $\mathbf{c}^l_t$ while a standard RNN has no cell state and is also able to memorize (even though less effective) via its cell's output $\mathbf{h}^l_t$. Cutting the blue and red connection also removes this temporal flow as $\mathbf{h}^l_{t-1}$ is never used.

Note that the cutting of recurrent connections is only done to gain more control over the memory leaking process to better define the memory time span. It is not intended to gain performance for the presented task. The research question could be generalized to exploring temporal information used in RNNs with LSTM-like cells.

4. Experiments

4.1. Experimental setup

All experiments were done on two speaker mixtures using the corpus introduced in [10], which contains artificial mixtures created by mixing together single speaker utterances from the Wall Street Journal 0 (WSJ0) corpus. For every utterance a gain factor was randomly chosen between 0 and 5 dB and utterances were sampled at 8kHz. The length of the mixture was chosen equal to the shortest utterance in the mixture as to maximize the overlap. The training and validation sets contained 20,000 and 5,000 mixtures, respectively and were taken from the `si_tr_s` set of WSJ0. The test set contained 3,000 mixtures using 16 held-out speakers of the `si_dt_05` and `si_et_05` set. The decimal log-magnitude (floored at -300) of the STFT with a 32 ms window length and a hop size (w_{hop}) of 8 ms were used as features and were normalized with mean and variance, calculated over the whole training set.

The used network has two fully connected leaky bidirectional LSTM (leaky BLSTM) layers with 600 hidden units each and was trained with the Adam learning algorithm [18]. Curriculum learning was used [19], i.e. the network was presented an easier task before tackling the main task. Here, the network was first trained on 100-frame non-overlapping segments of the mixtures. This network was then used as initialization when training over the full mixture. Zero mean Gaussian noise with standard deviation 0.2 was applied to the training features to avoid local optima. Dropout was not used since it did not improve the results in the experiments. The validation loss was calculated 3 times per epoch and early stopping was applied when the validation loss increased for 9 consecutive times. For DC the embedding dimension was chosen at $D = 20$ and since the frequency dimension was $F = 129$, the total number of output nodes was $DF = 20 * 129 = 2580$. Performance for MSSS was measured on the average signal-to-distortion ratio (SDR) improvements on the test set, using the `bss_eval` toolbox [20]. All networks were trained using TensorFlow [21] and the code for all the experiments can be found here: `https://github.com/JeroenZegers/Nabu-MSSS`.

To obtain the i-vectors, the UBM and T were trained on

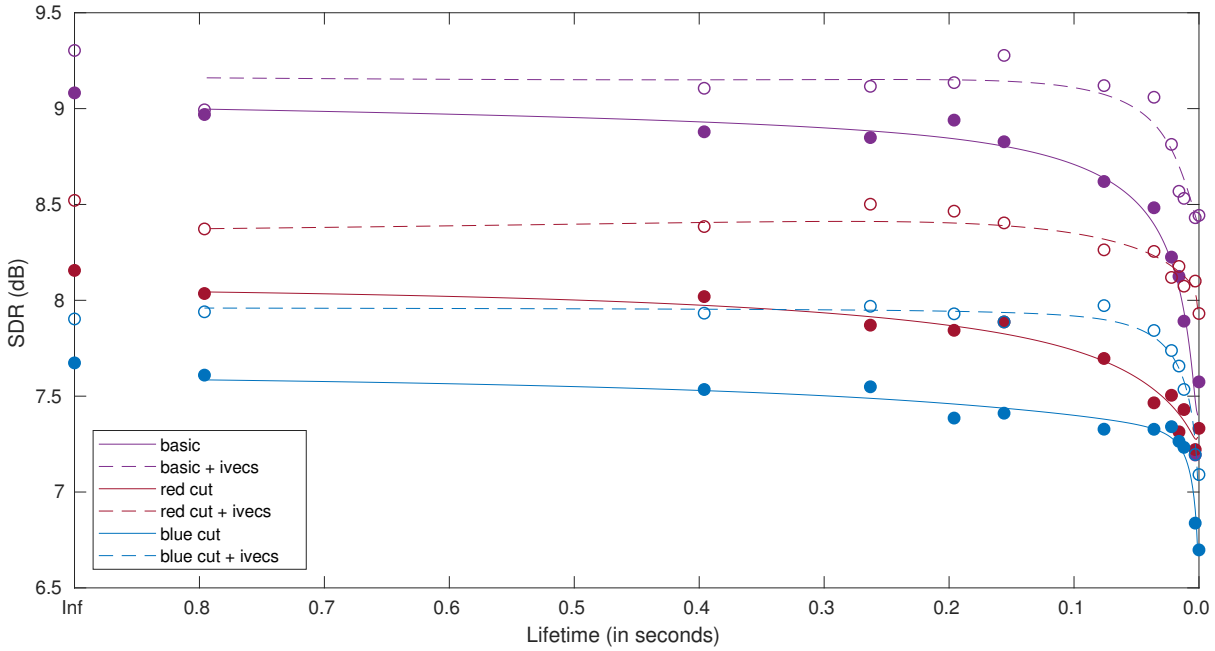

Figure 3: *MSSS results with LSTM leakage. The lifetime is given by* $\tau = -1/log(a) * w_{hop}$. *Experiments for* $\tau = 8.0\,s$ *and* $\tau = 1.6\,s$ *are not plotted but are used to fit the curves.*

development data from the `si_tr_s` set of Wall Street Journal 1 (WSJ1). 13-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) are used as features and a voice activity detector was used to leave out the silence frames. The UBM has 256 Gaussian mixtures and w is 10-dimensional, as was done in [9]. The i-vectors used in the experiments were obtained from the original single speaker utterances of WSJ0 but could also be obtained from speech signal reconstructions after source separation, as was done in [9]. The former was chosen since it provides a cleaner speaker representation. The MATLAB MSR Identity Toolbox v1.0 [22] was used to determine the UBM and T and to obtain the i-vectors.

4.2. Memory leakage

Networks were trained and tested for different values of a and for the different architectures as depicted by the dotted cuts in figure 2. Some networks were given the oracle i-vectors of both single speech signals used in the mixture. Results are shown in figure 3.

For all curves a rapid decrease in performance is found for $\tau < 100\,\text{ms}$. Here the leaky LSTMs might have difficulty with tracking the formants of the 2 speakers which span roughly 100 ms [23, 24]. For $\tau > 100\,\text{ms}$ the networks without i-vectors steadily keep increasing in performance, while the networks with i-vectors have a higher performance which is roughly constant. This seems to indicate that with a bigger time span, the networks without i-vectors manage better to find their own internal speaker representations. The networks with i-vectors don't need to find this speaker representation and therefor there performance stays constant for bigger time span. For no leakage ($\tau = \infty$) an i-vector is still a better speaker representation than the internal representation in the LSTM. When leakage is absolute ($\tau = 0$), performance is better when i-vectors are used, indicating that speaker information aids separation if no context is given and assignment is more consistent over frames.

Table 1: *The number of trainable parameters in the network in millions.*

	Basic	Red cut	Blue cut
$a > 0$	18.12	15.24	12.36
$a = 0$	15.08	12.21	10.77

Since the curves for the LSTMs with i-vectors are flat between $\tau = 100\,\text{ms}$ and $\tau = 300\,\text{ms}$ the network seems to give little importance to phonotactic and lexical information for the task of MSSS.

The shapes of the curves for all three types of LSTMs are quite similar, indicating that the bypass mechanisms of temporal information flow as described in section 3.3 are limited. The performance drop when cutting connections is expected as the model capacity is reduced by removing parameters (see table 1) and LSTM gates are more difficult to control.

Experiments were also done where no leakage was applied during training but only during testing. Results differed from those shown in figure 3. For example, no temporal information can be used if the LSTM with blue cuts and full leakage ($\tau = 0$) is used at test time. However, if the network was trained without leakage the SDR was 4.6 dB which is 2.0 dB below the performance obtained when it is also trained with full leakage, although both networks have no memory. This indicates it is indeed necessary to apply leakage also at training time to avoid mismatch.

5. Conclusions

Leakage was applied to the LSTM memory cell to determine the relevant memory time span for MSSS. A short time effect for formant tracking and a long time effect for speaker characterization were found. The same technique could be used to find the relevant time span for other tasks that use LSTMs.

6. References

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

[2] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[3] J. Chen and D. Wang, "Long short-term memory for speaker generalization in supervised speech separation," *The Journal of the Acoustical Society of America*, vol. 141, no. 6, pp. 4705–4714, 2017.

[4] Z. Meng, S. Watanabe, J. R. Hershey, and H. Erdogan, "Deep long short-term memory adaptive beamforming networks for multichannel robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 271–275.

[5] C. Kim, A. Menon, M. Bacchiani, and R. Stern, "Sound source separation using phase difference and reliable mask selection," 2018.

[6] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4470–4474.

[7] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The microsoft 2016 conversational speech recognition system," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5255–5259.

[8] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 1961–1970.

[9] J. Zegers and H. Van hamme, "Improving source separation via multi-speaker representations," in *Interspeech 2017*, 2017, pp. 1919–1923.

[10] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 31–35.

[11] Y. Isik, J. Le Roux, Z. Chen, S. Watanabe, and J. R. Hershey, "Single-channel multi-speaker separation using deep clustering," in *Interspeech 2016*, 2016, pp. 545–549.

[12] M. Kolbæk, D. Yu, Z. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," pp. 1901–1913, 2017.

[13] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 246–250.

[14] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, 1975.

[15] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[16] O. Glembek, L. Burget, P. Matějka, M. Karafiát, and P. Kenny, "Simplification and optimization of i-vector extraction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 4516–4519.

[17] C. Olah, "Understanding LSTM Networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015, [Online; accessed 23-March-2018].

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[20] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[22] S. O. Sadjadi, M. Slaney, and L. Heck, "Msr identity toolbox v1. 0: A matlab toolbox for speaker-recognition research," *Speech and Language Processing Technical Committee Newsletter*, vol. 1, no. 4, 2013.

[23] T. Gay, "Effect of speaking rate on diphthong formant movements," *The Journal of the Acoustical Society of America*, vol. 44, no. 6, pp. 1570–1573, 1968.

[24] N. Umeda, "Vowel duration in american english," *The Journal of the Acoustical Society of America*, vol. 58, no. 2, pp. 434–445, 1975.