PARALLEL LONG SHORT-TERM MEMORY FOR MULTI-STREAM CLASSIFICATION

Mohamed Bouaziz[1,2], *Mohamed Morchid*[1], *Richard Dufour*[1], *Georges Linarès*[1], *Renato De Mori*[1,3]

[1]LIA - University of Avignon (France)
[2]EDD - Paris (France)
[3]McGill University - Montreal, Quebec (Canada)

ABSTRACT

Recently, machine learning methods have provided a broad spectrum of original and efficient algorithms based on Deep Neural Networks (DNN) to automatically predict an outcome with respect to a sequence of inputs. Recurrent hidden cells allow these DNN-based models to manage long-term dependencies such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Nevertheless, these RNNs process a single input stream in one (LSTM) or two (Bidirectional LSTM) directions. But most of the information available nowadays is from multistreams or multimedia documents, and require RNNs to process these information synchronously during the training. This paper presents an original LSTM-based architecture, named Parallel LSTM (PLSTM), that carries out multiple parallel synchronized input sequences in order to predict a common output. The proposed PLSTM method could be used for parallel sequence classification purposes. The PLSTM approach is evaluated on an automatic telecast genre sequences classification task and compared with different state-of-the-art architectures. Results show that the proposed PLSTM method outperforms the baseline n-gram models as well as the state-of-the-art LSTM approach.

Index Terms: long short-term memory, sequence classification, stream structuring

1. INTRODUCTION

Recently, automatic sequence classification became an ubiquitous problem, having then encountered a high research interest [1, 2, 3]. This is due to the need to structure knowledge as a set of dependent localized information alongside with the new computer capabilities to efficiently process large amount of data. Among the recent methods employed to structure these sequences, the machine learning domain provides a set of high-level representations well adapted to automatic sequence classification based on Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN) [4] or Recurrent Neural Networks (RNN) [5].

RNN architectures such as Long Short-Term Memory (LSTM) [6] and Bidirectional LSTM (BLSTM) [7] have gained a particular attention in different domains and tasks including sentence [8] or successive images [9] processing. In speech recognition [10, 11, 12], these LSTM models exploit the contextual information whenever speech production or perception is influenced by emotion, strong accents, or background noise. The most effective use of RNNs for sequence classification is to combine the RNNs with Hidden Markov Models (HMMs) in a hybrid approach [13, 14]. Nonetheless, RNNs or RNN-HMM could not be directly employed for sequence classification using multiple inputs from synchronous streams such as TV shows coming from different channels. Indeed, RNNs can only be trained to make a set of elements labeled in a single stream of input information.

In this paper, we introduce an original multistream neural network architecture, called Parallel LSTM (PLSTM), that simultaneously takes into account different synchronous streams in order to automatically classify this multistream sequence. To evaluate the effectiveness of the proposed PLSTM multistream neural network architecture, experiments are carried out on the LIA's Electronic Program Guide (EPG) dataset containing 3 years of TV programs from 4 different channels. The PLSTM performance is compared with the LSTM state-of-the-art approach as well as a classic n-gram approach considered as the baseline. Our PLSTM approach is an important step for sequence classification since it can be applied to any set of synchronous sequences.

Section 2 proposes an overview of a couple of RNN architectures. Section 3 presents the proposed PLSTM. The experimental protocol and the discussion on the results are presented in Section 4 and 5 respectively. Finally, Section 6 concludes this work and gives some interesting perspectives.

2. RECURRENT NEURAL NETWORKS

This section introduces the state-of-the art concepts of two recurrent neural networks: LSTM and BLSTM.

2.1. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [6] networks are a special case of Recurrent Neural Networks (RNNs) [5]. The goal of this architecture is to create an internal cell state of the

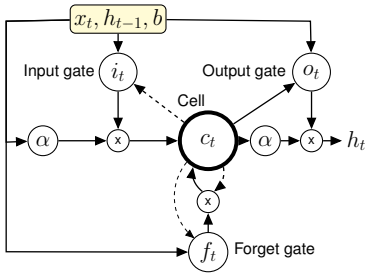

Fig. 1. Long Short-Term Memory (LSTM) cell. Dashed arrows correspond to connections with time-lag $(t-1)$. α input/output activation function is usually $\tanh$.

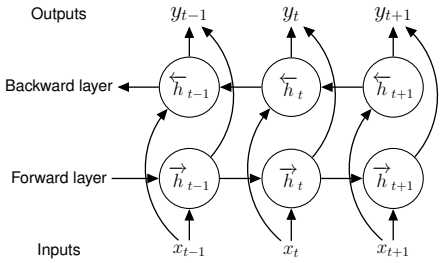

Fig. 2. Bidirectional Recurrent Neural Network (BRNN).

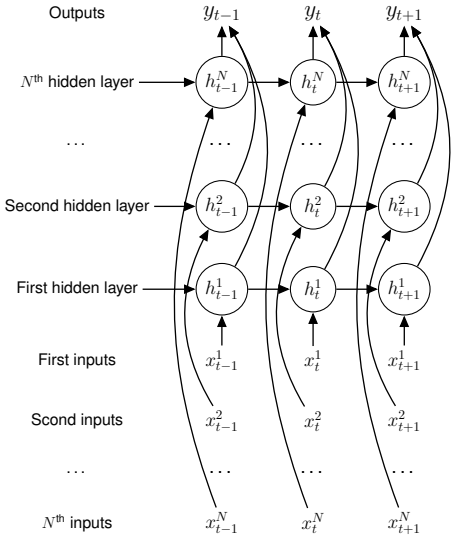

Fig. 3. Parallel Long Short-Term (PLSTM) neural network.

network which allows it to exhibit dynamic temporal behavior. This internal state allows the RNN to process arbitrary sequences of inputs such as sequences of words [8] for language modeling, time series [1]...The RNN takes as input a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ and computes the hidden sequence $\mathbf{h} = (h_1, h_2, \ldots, h_T)$ as well as the output vector $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ by iterating from $t = 1$ to T:

$$h_t = \mathcal{H}(\mathbf{W}_{xh}x_t + \mathbf{W}_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = \mathbf{W}_{hy}h_t + b_y \quad (2)$$

where T is the total number of sequences; $\mathbf{W}_{xh}$ are the weight matrices between the input layers **x** and **h** and so on; b is a bias vector, and $\mathcal{H}$ is the composite function. [6] shows that LSTM networks outperform RNNs for finding long range context and dependencies. The LSTM composite function $\mathcal{H}$ forming the LSTM cell with peephole connections [15] is presented in Figure 1 and defined as:

$$i_t = \sigma(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{W}_{ci}c_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(\mathbf{W}_{xf}x_t + \mathbf{W}_{hf}h_{t-1} + \mathbf{W}_{cf}c_{t-1} + b_f) \quad (4)$$

$$c_t = f_t c_{t-1} + i_t \tanh(\mathbf{W}_{xc}x_t + \mathbf{W}_{hc}h_{t-1} + b_c) \quad (5)$$

$$o_t = \sigma(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{W}_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t \tanh(c_t) \quad (7)$$

where i, f and o, are respectively the input, forget and output gates, and c the cell activation vector with the same size than the hidden vector h. The weight matrices $\mathbf{W}$ from cell c to gates i, f and o, are diagonal, and thus, an element e in each gate vector receives only the element e from the cell vector. Finally, σ is the logistic sigmoid function.

2.2. Bidirectional Long Short-Term Memory (BLSTM)

LSTM networks use only the previous context to predict the next segment for a given sequence. Bidirectional RNN (BRNN) [16], presented in Figure 2, can process both directions with two separate hidden layers (one for each direction). This type of RNN feeds to a same output layer fed forwarded inputs through the two hidden layers. Therefore, the BRNN computes both *forward* hidden sequence $\overrightarrow{\mathbf{h}}$ and *backward* sequence $\overleftarrow{\mathbf{h}}$ as well as the output vector **y**, by iterating $\overrightarrow{\mathbf{h}}$ from $t = 1$ to T, and $\overleftarrow{\mathbf{h}}$ from $t = T$ to 1:

$$\overrightarrow{h}_t = \mathcal{H}(\mathbf{W}_{x\overrightarrow{h}}x_t + \mathbf{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \quad (8)$$

$$\overleftarrow{h}_t = \mathcal{H}(\mathbf{W}_{x\overleftarrow{h}}x_t + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \quad (9)$$

$$y_t = \mathbf{W}_{\overrightarrow{h}y}\overrightarrow{h}_t + \mathbf{W}_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \quad (10)$$

By replacing the BRNN cells with LSTM cells, the Bidirectionnal LSTM (BLSTM) [7] is obtained. The BLSTM allows to exhibit long range context dependencies and takes advantage from the two directions structure. The output vector **y** is processed by evaluating simultaneously the two directions hidden sequences by computing the composite function $\mathcal{H}$ in the forward ($\overrightarrow{\mathbf{h}}$) and backward ($\overleftarrow{\mathbf{h}}$) directions.

3. PARALLEL LONG SHORT-TERM MEMORY (PLSTM)

The BRNN neural architecture uses the same sequence **x** as an input for both forward and backward directions, which is useful for information from a single stream. The paper proposes an original neural network, called Parallel RNN (PRNN) and presented in Figure 3, that takes advantage from the BRNN structure in a multistream context. By replacing the PRNN cells with LSTM cells, the proposed Parallel LSTM (PLSTM) is obtained.

The original PLSTM architecture corresponds to the PRNN description by replacing the $\mathcal{H}$ function with the LSTM composite function. PLSTM differs from the classical BLSTM by feeding forward, not a shared sequence, but different input vectors through a dedicated hidden layer $\mathbf{h}^n$ for each input vector $\mathbf{x}^n$. Moreover, BLSTM employs only 2 hidden layers due to its bidirectional concept while PLSTM can use multiple ones. The input sequences are considered independent and require to be mapped in homogeneous separate subspaces (**W** matrix from input **x** to hidden **h** spaces). Therefore, a single LSTM containing concatenated inputs from different independent sequences is not theoretically suitable for finding out a common homogeneous subspace to map heterogeneous input representation of parallel sequences.

Thus, for each n^{th} stream ($1 \leq n \leq N$), the PLSTM takes the input sequence $\mathbf{x}^n = (x_1^n, x_2^n, \ldots, x_T^N)$ and computes the hidden sequence $\mathbf{h}^n = (h_1^n, h_2^n, \ldots, h_T^N)$ and the output vector **y** by iterating from $t = 1$ to T.

$$h_t^N = \mathcal{H}(\mathbf{W}_{x^N h^N} x_t^N + \mathbf{W}_{h^N h^N} h_{t-1}^N + b_h^N) \quad (11)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

$$h_t^2 = \mathcal{H}(\mathbf{W}_{x^2 h^2} x_t^2 + \mathbf{W}_{h^2 h^2} h_{t-1}^2 + b_h^2) \quad (13)$$

$$h_t^1 = \mathcal{H}(\mathbf{W}_{x^1 h^1} x_t^1 + \mathbf{W}_{h^1 h^1} h_{t-1}^1 + b_h^1) \quad (14)$$

$$y_t = \sum_{n=1}^{N} \mathbf{W}_{h^n y} h_t^n + b_y \quad (15)$$

where N is the number of streams. In our experiments, the output vector **y** takes advantage of the N channels to predict the telecast's genre for one given channel n ($1 \leq n \leq N$). Therefore, PLSTM feeds forward separate sequences in order to predict a label and codes internal hidden structures between the parallel hidden sequences. [7] introduces the BLSTM with Back Propagation Trough Time (BPTT) algorithm [17] for training. For our proposed PLSTM architecture, the training takes place over N input sequences:

Forward Pass: feeds all input data for the sequences into the PLSTM and determines the predicted outputs.

- Do forward pass for each of the N forward states.
- Do forward pass for output layer.

Backward Pass: processes the error function derivative for the sequences used in the forward pass.

- Do backward pass for output neurons.
- Do backward pass for forward states.

Updating Weights

4. EXPERIMENTAL PROTOCOL

Multistream sequence classification is evaluated with the proposed PLSTM architecture (2 and 4 parallel sequences) as well as the classic LSTM network on an automatic TV show genre labeling task. Two n-gram based models (baseline) are also considered for fair comparison. Next sections describe the dataset, the genre sequence classification as well as the neural networks settings.

4.1. Multichannel EPG dataset

The Electronic Program Guide (EPG) dataset is extracted from 4 French TV channels (M6, TF1, France 5 and TV5 Monde) for 3 years, from January 2013 to December 2015. M6 channel is used in our experiments as the output stream. Data from *2013* and *2014* are merged and split into the *training* (70%) and *validation* (30%) datasets using a *stratified shuffle split* [18] in order to preserve the same percentage of samples of each class in the output of both folds, while the *2015* dataset is kept for testing. In order to guarantee a clean experimental environment, labels (*i.e.* genres) that are absent at least in one of the three folds were removed. Doing so allows us to have equivalent datasets in terms of labels vocabulary. Table 1 shows the genres distribution for M6, the chosen output channel.

Genres	Training	Validation	Test
Weather	2,691	1,153	1,712
Fiction	1,890	810	1,478
News	913	392	679
Other magazine	981	421	466
Music	461	197	340
Teleshopping	421	180	308
TV game show	476	204	287
Cartoon	361	155	205
Other	277	119	133
Reality TV	83	36	76
Documentary	29	13	14
Total	8,107	3,680	5,698

Table 1. Genres Distribution for train, validation and test sets in M6 channel output.

4.2. Genre Prediction Experiments

For a given input history sequence (composed of the n previous telecast genres), a genre label representing the next M6's

telecast is output. The size of the genre sequences (n) varies from 1 to 4. Then, three input configurations are employed. **Mono-channel input:** only M6 history sequences for a baseline *n-gram* experiment (with a *statistical language model* from the SRILM toolkit [19]) and a straightforward *LSTM* model. **Bi-channel input:** both M6 and TF1 channel histories are employed as input for *P2LSTM* (PLSTM with two parallel streams as a BLSTM with forward-forward directions and separate inputs). The aim of this experiment is to move up the context's information using a similar and rival channel, the two being generalist channels. **Multichannel input:** History of each of the 4 streams (*i.e.* channels) is used as input for *4n-gram* and P4LSTM experiments (PLSTM with 4 parallel streams).

4.3. Neural Networks Setup

The classical LSTM, and the proposed P2LSTM and P4LSTM, are composed with 3 layers: input layer **x** of size varying from 1 to 4, a hidden layer **h** of size 80 for all LSTM-based models and an output layer **y** with a size equals to the number of different possible TV genres (11). The Keras library [20], based on Theano [21] for fast tensor manipulation and CUDA-based GPU acceleration, has been employed to train neural networks on an Nvidia GeForce GTX TITAN X GPU card. The training times, detailed in Table 2 for all models, match with the sequence size of all models. Indeed, even with the most time-consuming configuration, namely P4LSTM with 4 elements history, the training does not last more than 25 minutes.

Sequence size	**1**	**2**	**3**	**4**
n-gram	1	1	1	1
4n-gram	2	5	17	51
LSTM	51	146	319	362
P2LSTM	259	473	485	439
P4LSTM	536	923	844	1,386

Table 2. Training times (in seconds) of models employed during the experiments for different telecast genres sequence sizes.

5. RESULTS AND DISCUSSION

Table 3 shows the overall results, in terms of the standard F1 metric related to the genre prediction task outputs, using each method and for different stream sequence sizes from 1 to 4.

Seq. size	**n-gram**	**4n-gram**	**LSTM**	**P2LSTM**	**P4LSTM**
1	19.07	59.39	11.60	47.46	47.24
2	51.38	58.35	34.64	54.17	59.54
3	57.41	57.10	50.74	58.69	59.92
4	56.87	57.26	56.47	58.67	**60.81**

Table 3. F1-score (%) of each n-gram and LSTM models.

5.1. N-gram based models

The multi-channel 4n-gram model outperforms the simple n-gram one for each of the different 4 genre sequence configurations except for 3 sized history. *4n-gram* method reaches around 59% of F-score using 1 sized sequences against near 57% for mono-channel n-gram using its best history configuration. The observed results confirm the interest of using multiple streams to predict the next telecast's genre for a specified channel.

5.2. LSTM and PLSTM

One can figure out from Table 3 that mono-channel LSTM does not even outperform the mono-channel n-gram experiment. P4LSTM obtains the best result with an F1 score close to 61% using a sequence of size 4. In order to analyze these results, the Error Rates (ER) are also presented in Table 4. The overall F1 scores are different from those related to the ER. For example, at its best configuration of a 4 sized sequence, P4LSTM error rate reaches about 23.5%, which corresponds to a correct rate of 76.5% against an F1-measure of only 61%. Moreover, although the range between the lowest and the highest values is almost 12 points for ER, it is only near 4 points for F1-measure. The reason of this is that the F1-metric may be not suitable for the task due to the labels imbalance with different numbers of genre occurrences varying from 14 to 1,712 in the test set.

Seq. size	**n-gram**	**4n-gram**	**LSTM**	**P2LSTM**	**P4LSTM**
1	51.36	30.29	60.13	36.68	34.35
2	39.06	28.99	48.99	29.38	25.08
3	31.71	29.64	35.73	28.85	24.76
4	35.43	30.5	29.59	28.27	**23.52**

Table 4. Error rates (ER) observed for each n-gram and LSTM models for different sequence sizes.

5.3. Discussion

Confusion matrices of 4n-gram and P4LSTM methods are shown in Tables 5 and 6 to point out benefits of the proposed PLSTM model.

Weather	1282	**65**	0	**323**	11	4	17	0	5	2	3
Fiction	**292**	**904**	0	**177**	35	9	15	26	16	0	4
News	2	12	636	1	27	0	0	0	1	0	0
Other mag.	**69**	**40**	3	296	10	2	6	0	29	6	5
Music	**16**	**35**	17	5	219	0	3	2	43	0	0
Teleshop.	**45**	1	0	1	0	**246**	0	0	15	0	0
TV game sh.	2	7	0	19	0	0	**245**	0	6	8	0
Cartoon	0	32	6	0	9	***102***	0	56	0	0	0
Other	10	10	0	**25**	1	2	15	0	**66**	2	2
Reality TV	**20**	14	0	9	0	0	4	0	19	**9**	1
Docum.	**4**	3	0	**4**	0	0	0	0	2	0	**1**

Table 5. Confusion matrix for the 4n-gram output: labels are shown according to their decreasing frequency as in Table 1.

Weather	**1624**	**38**	0	**46**	0	2	0	0	2	0	0
Fiction	**375**	861	0	**188**	18	0	18	16	2	0	0
News	4	0	**669**	1	5	0	0	0	0	0	0
Other mag.	**114**	**34**	0	**306**	6	0	3	0	3	0	0
Music	15	18	0	1	**297**	0	7	2	0	0	0
Teleshop.	**49**	15	0	0	0	244	0	0	0	0	0
TV game sh.	**49**	14	0	14	11	0	199	0	0	0	0
Cartoon	**25**	**28**	1	0	0	4	0	**147**	0	0	0
Other	**70**	3	0	**48**	0	1	0	0	11	0	0
Reality TV	**35**	0	0	**23**	0	0	14	0	4	0	0
Docum.	**8**	0	0	**5**	0	0	1	0	0	0	0

Table 6. Confusion matrix for the P4LSTM output: labels are shown according to their decreasing frequency as in Table 1.

It is worth emphasizing that most of the missed instances in all systems are wrongly labeled as one of the two most frequent classes, *Weather* and *Fiction*, as well as the *Other Magazine* genre (some examples are in green cells). *False positives* are more recurrent in some small classes such as *Other Magazine* than in the relatively more frequent class *News*. The reason is that *News* is a well defined genre occurring mostly at the same time each day unlike *Other Magazine* genre that encompasses various telecasts that are broadcast at several and irregular daytime. *Teleshopping* shows are often broadcast at nearly the same time of morning than *Cartoons* and affects dramatically the performance of the 4n-gram model in this context (cf. underlined italic cell in Table 5). Finally, the confusion matrix of P4LSTM experiment shows that this system fails more dramatically to predict the least frequent genres *Others*, *Reality TV*, and *Documentary*. For example, for the two least frequent genres, *Reality TV* and *Documentary*, respectively none of the 76 and the 14 instances were correctly found. This leads to a *precision* of 0 which penalizes the average precision and then the overall F-score.

Seq. size	n-gram	4n-gram	LSTM	P2LSTM	P4LSTM
1	23.30	70.32	14.18	58.00	57.74
2	59.47	68.04	42.34	66.21	72.77
3	66.71	66.55	62.01	71.74	73.23
4	65.67	66.77	69.01	71.71	**74.32**

Table 7. F1 score (%) of n-gram and LSTM models, the two least frequent genres *Reality TV* and *Documentary* not being included.

In order to evaluate the impact of the least frequent genres on the F1 measure, especially on the three LSTM based systems, we also reported on Table 7 the F1 results on the same outputs of the experiments of Table 3 by excluding the two least frequent genres from the averages of precision and recall (*Reality TV* and *Documentary*). The state-of-the-art mono-channel LSTM performances gradually become closer and closer to the multichannel n-gram model ones (cf. Figure 4) when the size of sequences moves up with an F1 score of 69%. Therefore, LSTM-based models require longer sequences to learn long term dependencies than the 4n-gram

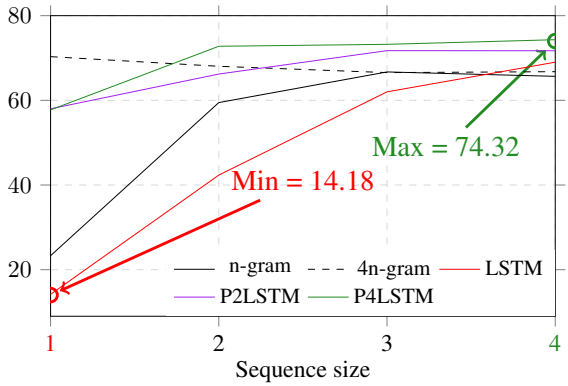

Fig. 4. F1 score for n-gram and LSTM models, the two least frequent genres *Reality TV* and *Documentary* not being included.

methods.

Overall, the results of the PLSTM detailed in Table 4 and Figure 4, demonstrate the benefits obtained at least for history sequences longer than 2 genres with an F1 score greater than 71%.

Regarding multichannel P4LSTM approach, the highest performance reaches an F1-measure of about 74% using 4 sized sequences with a gain of about 3 and 4 points compared respectively to P2LSTM and 4n-gram model best performances.

6. CONCLUSION

The paper proposes an original Long Short-Term Memory (LSTM) based neural network architecture for automatic classification of multistream sequences called PLSTM. PLSTM is evaluated during a telecast genre prediction task and the observed results show that the proposed PLSTM is efficient when the size of sequences is large enough with a gain of more than 8 points compared to classical n-gram model, and about 5 and 3 points respectively compared to LSTM and P2LSTM. Future works will apply this promising multistream neural network architecture to Spoken Language Understanding tasks such as topic extraction, keyword spotting and Part-of-Speech tagging.

7. REFERENCES

[1] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," in *Artificial Neural NetworksICANN 2001*. Springer, 2001, pp. 669–676.

[2] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th International ACM SIGIR Confer-*

ence on Research and Development in Information Retrieval. ACM, 2015, pp. 959–962.

[3] M. Huang, Y. Cao, and C. Dong, "Modeling rich contexts for sentiment classification with lstm," *CoRR*, vol. abs/1605.01478, 2016.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

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

[7] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[8] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling." in *INTERSPEECH*, 2012, pp. 194–197.

[9] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[10] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*. Springer, 2005, pp. 799–804.

[11] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Artificial Neural Networks–ICANN 2007*. Springer, 2007, pp. 220–229.

[12] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, and R. Cowie, "Abandoning emotion classes-towards continuous emotion recognition with modelling of long-range dependencies." in *INTERSPEECH*, vol. 2008. Citeseer, 2008, pp. 597–600.

[13] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, 2012, vol. 247.

[14] Y. Bengio, "Markovian models for sequential data," *Neural computing surveys*, vol. 2, no. 1049, pp. 129–162, 1999.

[15] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

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

[17] M. Schuster, "On supervised learning from sequential data with applications for speech recognition," *Daktaro disertacija, Nara Institute of Science and Technology*, 1999.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] A. Stolcke *et al.*, "Srilm-an extensible language modeling toolkit." in *INTERSPEECH*, vol. 2002, 2002, p. 2002.

[20] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[21] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.