INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Small-Footprint Magic Word Detection Method Using Convolutional LSTM Neural Network

Taiki Yamamoto[1], *Ryota Nishimura*[1], *Masayuki Misaki*[2], *Norihide Kitaoka*[1]

[1]Department of Advanced Technology and Science, Tokushima University, Tokushima, Japan
[2]Panasonic Corporation, Osaka, Japan

yamat_0129@outlook.jp

Abstract

The number of consumer devices which can be operated by voice is increasing every year. Magic Word Detection (MWD), the detection of an activation keyword in continuous speech, has become an essential technology for the hands-free operation of such devices. Because MWD systems need to run constantly in order to detect Magic Words at any time, many studies have focused on the development of a small-footprint system. In this paper, we propose a novel, small-footprint MWD method which uses a convolutional Long Short-Term Memory (LSTM) neural network to capture frequency and time domain features over time. As a result, the proposed method outperforms the baseline method while reducing the number of parameters by more than 80%. An experiment on a small-scale device demonstrates that our model is efficient enough to function in real time.

Index Terms: keyword spotting, convolutional neural network, recurrent neural network, convolutional LSTM, small footprint

1. Introduction

Automatic speech recognition (ASR) systems are being used more widely on consumer devices such as smartphones, smart speakers and computer operating systems. Many of these devices can be controlled by voice only, without touching the screen or performing other operations. Some systems, including Google Assistant and Apple's Siri, start up when a specific "magic word" is detected, allowing the devices to be operated verbally, without touching them. Magic Word Detection (MWD), also known as "wake-up word detection", is the technology which makes this possible, and it has become essential for the hands-free operation of electronic devices.

MWD systems attempt to detect these magic words from continuous audio streams of speech, thus MWD systems function as discriminator, determining whether or not a user has uttered the magic word. Currently, most MWD systems use Neural Networks (NNs) as the discriminator. Deep NN (DNN) systems have been shown to outperform Hidden Markov Models in speech detection tasks[1, 2], so NN systems have become the de facto standard for MWD. However, these high-performance NNs contain many layers and units, and since MWD systems need to run continuously to detect Magic Words at any time, this puts a burden on the limited resources of such devices. To address this problem, many studies have focused on limiting the memory and computational resources needed for MWD in order to realize small-footprint systems [2, 3, 4].

Feed-forward DNNs are one common word detection approach used, as in [2]; however, feed-forward DNNs are unable to capture input context in the time or frequency domains [5]. Convolutional Neural Networks(CNNs) have also been explored, with the goal of capturing such local connectivity using shared weights [3]. However, CNNs cannot capture context over entire frames without wide filters or great depth. Recurrent Neural Networks (RNNs) have also been studied for MWD [6, 7], but like DNNs, RNNs also have difficulty capturing the context of the input in the frequency domain. As a result, RNNs are unable to achieve a low False Alarm (FA) rate at higher levels of detection accuracy. Therefore, Convolutional RNNs (CRNNs) have been explored for MWD [4]. CRNNs have structures which combine convolution layers with recurrent layers, allowing them to achieve low FA rate at high rates of detection accuracy [4]. In a recent study [8], further improvement in accuracy was achieved by applying an attention mechanism with RNNs and CRNNs.

Inspired by [9], we propose various NN models using a Convolutional LSTM (CLSTM) [10], an approach shown to improve performance for Large Vocabulary Continuous Speech Recognition (LVCSR) [9]. CLSTMs have a few advantages over RNNs: CLSTMs can capture spatio-temporal changes, and the number of parameters needed for CLSTMs is smaller than the number need by LSTMs. In this paper, we explore a CLSTM architecture suitable for small-footprint MWD systems. We show that CLSTMs can outperform baseline CRNNs, while reducing the number of required parameters by more than 80%.

2. Network Architectures

In this section, we explain the architectures of LSTM and Convolutional LSTM networks.

2.1. LSTM

As demonstrated in [11], the simple architecture of RNNs makes them incapable of capturing long-term relationships within input. Long Short-Term Memory (LSTM) networks work out these relationships using cells and gates [12, 13]. The cells use gate structures to choose the necessary information within current and recurrent states, allowing the LSTM to handle relatively long sequences of data. At time frame t ($t = 1, 2, ..., T$), when current input vector x_t, recurrent input vector h_{t-1}, and cell state c_{t-1} are given, an LSTM update can be formulated as:

$$\begin{aligned}
i_t &= \sigma(w_{xi}x_t + w_{hi}h_{t-1} + w_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(w_{xf}x_t + w_{hf}h_{t-1} + w_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} \\
&\quad + i_t \circ \tanh(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \qquad (1) \\
o_t &= \sigma(w_{xo}x_t + w_{ho}h_{t-1} + w_{co} \circ c_{t-1} + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}$$

where i_t, f_t, c_t and o_t denote the input gate, forget gate, cell update gate, and output gate, respectively, and $\circ$ denotes the Hadamard product.

http://dx.doi.org/10.21437/Interspeech.2019-1662

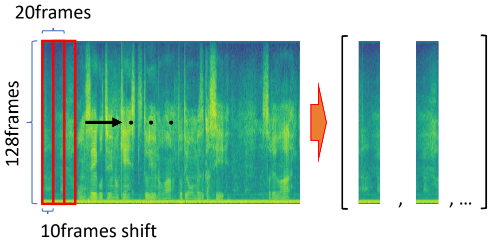


Figure 1: *Image Segmentation: We divided the sequence of input frames into segments (indicated with red boxes).*

2.2. Convolutional LSTM

Convolutional LSTMs (CLSTMs) [10] use previous and current spatial information, such as current matrix X_t and recurrent matrix H_{t-1}. This allows the CLSTM to propagate spatial information to the next time frame, making it possible to capture temporal changes in the input matrix. At time frame t ($t = 1, 2, ..., T$), when current input matrix X_t, recurrent input matrix H_{t-1}, and cell states C_{t-1} are given, a CLSTM can be formulated as:

$$\begin{aligned} I_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\ F_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\ C_t &= F_t \circ C_{t-1} \\ &\quad + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \qquad (2) \\ O_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_{t-1} + b_o) \\ H_t &= O_t \circ \tanh(C_t) \end{aligned}$$

where I_t, F_t, C_t and O_t denote the input gate, forget gate, cell update gate, and output gate, respectively, and $\circ$ and $*$ denote the Hadamard product and the convolutional operation, respectively. In comparison with Eq.(1), weighted calculations are simply replaced by convolutions. Also, note that the matrix size should remain unchanged between the CLSTM input and output. Thus, we have to prevent the size of the matrix H_T from changing since the CLSTM uses H_t for the subsequent time frame $(t + 1)$.

2.3. Advantages of Convolutional LSTM

An LSTM is a kind of fully-connected network of feed-forward DNNs across time, as mentioned in Section1, so they are unable to capture input context in either the time or frequency domains [5]. In contrast, CLSTMs are able to capture input context in both the time and frequency domains. We believe this will allow a CLSTM to effectively capture input context expanded complexly in both the time and frequency domains over the time frames or segments. Thus, we expect a CLSTM to achieve better results on MWD tasks.

3. Experimental Setup

In this section we explain our experimental settings, such as the data set, model architectures and training settings.

3.1. Dataset

We constructed an original data set for this study for training and testing. We set the Japanese word "ごえもん" (Romanization: "goemon") as our Magic Word, which could be used to activate a home robot to perform tasks when it detects this word in human speech. Goemon is a popular ninja or bandit character name in Japan. We created five suitable scenarios involving the use of the Magic Word, each of which consisted of a dialog between two subjects, A and B. During recording, we combined the subjects' monaural audio streams, which were captured by lapel microphones, into a stereo stream using a PCM recorder. The quantization and sampling rate of the recorded speech data were set to 24 bit and 48 kHz, respectively. After recording, the speech data were converted to 16 bit and 16 kHz and split into monaural speech data. We collected a total of 200 samples of monaural data from 20 pairs of subjects (40 subjects in total). Next, we converted the speech into log power spectrograms using FFT and a Hamming window. Window size and shift width were set at 256 points (16 ms) and 160 points (10 ms), respectively. We then applied segmentation in order to create a matrix of stacked frames, as shown in Fig.1. The segment size and shift width were set to 20 frames (206 ms) and 10 frames (100 ms), respectively. Finally we added Positive or Negative labels to each segment; Positive (1) for segments containing the Magic Words, and Negative (0) for all the others. Segments containing 70% period from the end of the Magic Word were labelled as Positive. As a result, we obtained 187,184 Negative segments(10.711 hours) and 500 Positive segments (103 seconds). This data set was then devided into training and test sets at a ratio of 9:1.

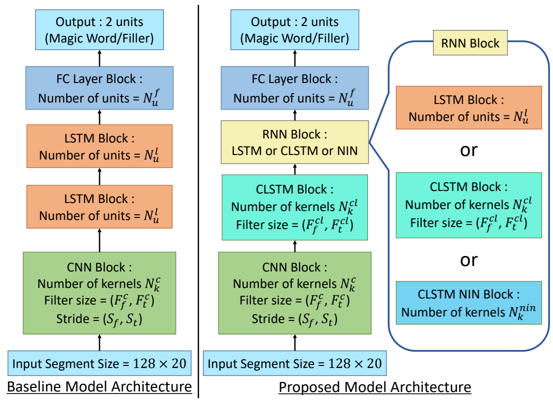


Figure 2: *Overview of the Baseline and Proposal models: The flowchart on the left shows the architecture of the Baseline model, while the flowchart on the right shows the architecture of the proposed model. The RNN Block in the Proposed model can be an LSTM or CLSTM or CLSTM-NIN.*

3.2. Model Architectures and Training Settings

We compared several variations of our models, all of which included a CNN block and stacked RNN blocks. The details of each variant's architecture are as shown in Tables 1. As in [3, 4], given the discrepancy in input dimensionality and the training data, we optimized all of the models using a limit on the number of parameters of 250,000 to allow for a fair comparison.

The baseline models (Table 1) used a CRNN architecture similar to the one proposed in [4]. In this paper, the baseline models are called $lstm\_lstm$ because the LSTM layers are stacked (the architecture shown on the left of Fig.2). According to [9], BiCLSTMs are unsuitable for real time processing; therefore, we did not use one. Our proposed models have uni directional CLSTM based architectures (the architecture shown on the right of Fig.2). Our RNN block can contain either an LSTM, CLSTM or CLSTM NIN. In Table 1, the notation $clstm\_lstm$ means the RNN block contains CLSTM followed by an LSTM, while $clstm\_clstm$ contains two stacked CLSTMs, and $clstm\_nin$ contains a CLSTM followed by a CLSTM NIN. A CLSTM NIN is a CLSTM with 1×1 kernel, and this layer functions as a Network-in-Network (NIN) [14].

Table 1: *Specifications of Baseline and Proposed Models: The kernel size of CNN Block was fixed ($F_f{}^c$, $F_t{}^c$) = (7, 5). Also, the kernel size and stride of CLSTM Block was fixed ($F_f{}^{cl}$, $F_t{}^{cl}$) = (3, 3) and ($S_f{}^{cl}$, $S_t{}^{cl}$) = (1, 1), respectively.*

	CNN Block		RNN Blocks		FC	# Params.	# Mul.
Baseline Model Name	$N_k{}^c$	$(S_f{}^c, S_t{}^c)$	$N_u{}^{l1}$	$N_u{}^{l2}$	$N_u{}^f$	-	-
$lstm\_lstm$	16	(3, 1)	40	40	80	248k	606k
$lstm\_lstm\_sml$	8	(3, 1)	10	10	20	30k	213k
Proposed Model Name	$N_k{}^c$	$(S_f{}^c, S_t{}^c)$	$N_k{}^{cl}$	$N_u{}^l$	$N_u{}^f$	-	-
$clstm\_lstm$	16	(3, 1)	12	122	245	247k	2.6M
$clstm\_lstm\_sml$	8	(3, 1)	8	25	50	30k	981k
-	$N_k{}^c$	$(S_f{}^c, S_t{}^c)$	$N_k{}^{cl1}$	$N_k{}^{cl2}$	$N_u{}^f$	-	-
$clstm\_clstm$	16	(3, 1)	16	16	520	247k	3.8M
$clstm\_clstm\_sml$	8	(3, 1)	8	8	84	30k	1M
-	$N_k{}^c$	$(S_f{}^c, S_t{}^c)$	$N_k{}^{cl}$	$N_k{}^{nin}$	$N_u{}^f$	-	-
$clstm\_nin$	16	(1, 1)	12	12	280	247k	4.5M
$clstm\_nin\_sml$	8	(1, 1)	8	4	62	30k	1.7M
$clstm\_nin\_2$	16	(1, 1)	12	24	133	247k	5.1M
$clstm\_nin\_2\_sml$	8	(1, 1)	8	8	25	30k	1.8M

An NIN is actually a multilayer perceptron rather than a CNN, which is able to express more complicated projections than typical CNNs. However, an NIN is the equivalent of a CNN with a 1×1 kernel. In our proposed model architecture, convolution is applied in a temporal direction, so the NIN is also adopted in a temporal direction. The $clstm\_nin\_2$ is larger model in the number of NIN kernel in $clstm\_nin$. In order to evaluate whether each model is effective for small-footprint MWD system, the sml variants were re-optimized by limiting the number of parameters of about 30,000.

We used ReLU in the CNN and FC layers in all models. Batch-normalization was applied to CNN block's output, followed by a max pooling layer with 4×2 kernel (stride size was the same as kernel size). In addition, in all of the proposed variants the last CLSTM or CLSTM NIN block is followed by a max pooling layer with 2×2 kernel (stride size was the same as kernel size). For example, if RNN block is an LSTM, the CLSTM was followed by a max pooling layer. All of the proposed models were able to obtain good results, and the number of parameters was reduced by following the last CLSTM or CLSTM NIN layer with a max pooling layer. Furthermore, Dropout [15] was applied to all of the RNN Blocks. Dropout is also effective for CNNs [15], so we also applied it to the CLSTM architectures. The Dropout rate for all of the RNN blocks was set to 0.25.

We used Chainer [16] for NN implementation, training and testing. All of the models were classification models, so the softmax function was applied to the output layer. The loss functions of all of the models, including the baseline model [4], were set using a Softmax cross entropy loss. We used Adam [17] as our optimization method. The learning rate was initialized at 0.001 and the batch size was set to 60. All models were trained using the back-propagation through time (BPTT) method. BPTT length was set to $T = 1.5$ sec.

4. Experimental Results

In this section, we explain and discuss our experimental results. We chose the best performing model in iterations and used it for our evaluation experiment. MWD performance was measured using a Reciever Operating Characteristic (ROC) curve. The vertical and horizontal axes represent the False Rejection rates (FRr) and False Alarms per hour (FAh), respectively. The lower the FRr per FA/h, i.e., the closer the curve is to the bottom-left of the graph, the better a model's Magic Word detection performance.

4.1. Comparison of all models

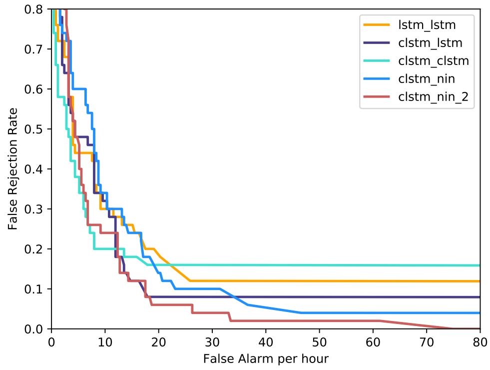
ROC curves for all of the models described in Table 1 are shown in Fig.3. As shown in Fig.3 (A), $clstm\_nin\_2$ was the best performing model, followed by $clstm\_lstm$ and $clstm\_lstm\_sml$. The $clstm\_clstm$ model did not obtain results as good as $clstm\_lstm$ or $clstm\_nin$, although the FRr of the $clstm\_clstm$ model was better than the baseline CRNN ($lstm\_lstm$) at the FAh ≤ 21. The FRr of our proposed $clstm\_nin\_2$ improved 57.5% from the FRr of $lstm\_lstm$ at the FAh = 25. Regarding the small models, as shown in Fig.3 (B), $clstm\_lstm\_sml$ and $clstm\_nin\_2\_sml$ out-performed the other models. Therefore, when comparing the performance of the baseline and CLSTM models, based on our results we believe that the CLSTM models outperform the models using only LSTM.

4.2. Improvement of CLSTM NIN model performance

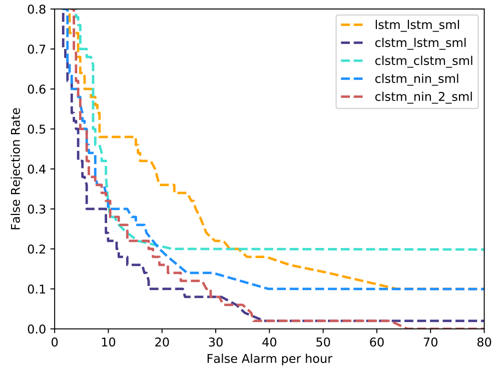
When comparing $clstm\_lstm$ and $clstm\_nin$, we can see that $clstm\_lstm$ has a much larger number of the recurrent units. So, to compare them more accurately, we adjusted the number of parameters to increase the weights in NIN ($clstm\_nin\_2$, $clstm\_nin\_2\_sml$), keeping the total number of parameters around the previous limits of 250k or 30k. As shown in Fig.3, when comparing $clstm\_nin$, $clstm\_nin\_sml$, $clstm\_nin\_2$ and $clstm\_nin\_2\_sml$, we can see that $clstm\_nin\_2$ and $clstm\_nin\_2\_sml$ achieved superior performance. Therefore, we think that a structure with many the number of weights is more effective for the CLSTM NIN following CLSTM.

4.3. Impact of CLSTM on reducing footprint

We ranked all of the models by FRr at 25 FA/h, and the five, top-perfoming models are listed in Table 2. ROC curves for the top five models are shown in Fig.4. As shown in Tables 1 and 2, models $clstm\_lstm\_sml$ and $clstm\_nin\_2\_sml$ achived higher performance than $lstm\_lstm$, while reducing the number of parameters by more than 80%. In addition, we compared sml models in order to disprove the suspicion that the



(A) ROC curves of the normal models



(B) ROC curves of the small models

Figure 3: *ROC curves for all models: (A) the normal models; (B) the small models.*

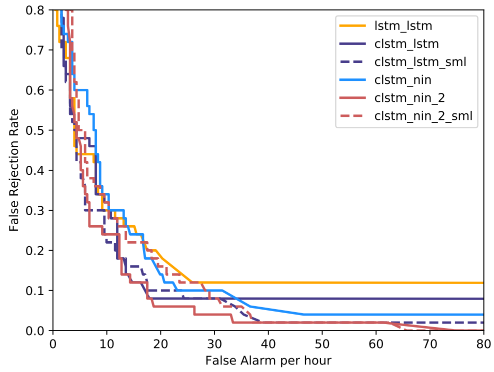


Figure 4: *Comparison of the top five curves + baseline curve*

Table 2: *Comparison of the top five models + baseline model (False Rejection rates at 25 False Alarms per hour)*

Model Name	FRr at 25 FAh
$lstm\_lstm$	13.2%
$clstm\_lstm$	7.8%
$clstm\_lstm\_sml$	7.8%
$clstm\_nin$	9.5%
$clstm\_nin\_2$	**5.6%**
$clstm\_nin\_2\_sml$	13.8%

proposed sml models performed well simply due to its small size. In fact, $lstm\_lstm\_sml$ did not outperform the proposed sml models, thus, we believe that CLSTM and CLSTM NIN would both be effective for small-footprint MWD systems. However, CLSTM-based models involve greater computational costs (due to more multiplication operations) than non-CLSTM-based models. Considering the tradeoff between the number of parametars and multiplication operations, we think it is more practical to use $clstm\_lstm\_sml$ model for MWD system.

4.4. Testing Models on a Small Device

As mentioned in Section 4.3, CLSTM-based models must perform a large number of multiplication operations, while MWD systems need to be able to operate in real time. For these reasons, we conducted additional experiments by running the top five models on a small-scale device, a Raspberry Pi 3 Model B+. We measured the average execution time per one segment needed to process one minute of speech. The results are shown in Table 3. It takes our system about 20 ms for waveform processing, so DNN processing should be accomplished within 80 ms. In Table 3, the models whose execution times are shown in bold (models $clstm\_lstm$, $clstm\_lstm\_sml$ and $clstm\_nin\_2\_sml$) were able to operate in real-time. We can also see that the $clstm\_lstm\_sml$ model was the most efficient, making it the best candidate for high precision MWD with real-time processing. Models $clstm\_nin$ and $clstm\_nin\_2$ had difficulty operating in real-time. However, as shown in Tables 1 and 3, although the number of multiplication required by $clstm\_nin\_2\_sml$ model was smaller than the number required by the $clstm\_lstm$, $clstm\_nin\_2\_sml$ model still required more execution time than the $clstm\_lstm$ model. In future work, it will be necessary to implement faster CLSTM models.

Table 3: *Average execution times per one segment needed by the top five models to process one minute of speech running on Raspberry Pi 3 Model B+ (CPU: BCM2837B0,4cores,1.4GHz).*

Model Name	Exec. Time
$clstm\_lstm$	**41ms**
$clstm\_lstm\_sml$	**39ms**
$clstm\_nin$	83ms
$clstm\_nin\_2$	86ms
$clstm\_nin\_2\_sml$	**63ms**

5. Conclusions

In this paper, we explored suitable CLSTM architectures for small-footprint MWD systems. Our proposed models combined, convolutional LSTMs with another LSTM or a Network-in-Network, enabling them to outperform the best performance of the baseline model while achieving a more than 80% reduction in the number of parameters required. In particular, the combination of a convolutional LSTM with a second LSTM with a small number of parameters allowed us simultaneously achieve a high level of performance, a small-footprint and real-time operation. As mentioned in Section 4.4, CLSTMs have the drawback of requiring a large number of multiplication operations. Despite this problem, some of our proposed models were able to operate in real-time on a small-scale Raspberry Pi 3 Model B+ device. Therefore, we believe that CLSTM-based models are effective for small-footprint MWD. In future work, we will adjust the hyper-parameters of CLSTMs to reduce the number of multiplication operations. As another goal, we would like to develop a model that allows users to change their Magic Word arbitrarily.

6. References

[1] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE, 1990, pp. 129–132.

[2] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks." in *ICASSP*, vol. 14. Citeseer, 2014, pp. 4087–4091.

[3] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *INTERSPEECH2015 Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *INTERSPEECH 2017*, pp. 1606–1610, 2017.

[5] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=303568.303704

[6] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proceedings of the 17th International Conference on Artificial Neural Networks*, ser. ICANN'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 220–229. [Online]. Available: http://dl.acm.org/citation.cfm?id=1778066.1778092

[7] A. H. Chris Lengerich, "An end-to-end architecture for keyword spotting and voice activity detection," in *NIPS 2016 End-to-end Learning for Speech and Audio Processing Workshop*. NIPS, 2016.

[8] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," *arXiv preprint arXiv:1803.10916*, 2018.

[9] M. Fujimoto and H. Kawai, "Comparative evaluations of various factored deep convolutional rnn architectures for noise robust speech recognition," *ICASSP 2018*, pp. 4829–4833, 2018.

[10] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[11] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

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

[14] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. 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.

[16] "Chainer," https://docs.chainer.org/en/v4.0.0/index.html#, last Access 14th-Oct-2018.

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