RESEARCH ARTICLE | MARCH 26 2024

Evaluation of learning rate training model on heart disease detection using LSTM FREE

Amrul Faruq ✉; Bellina Rahmamaulida Adeyani; Lailis Syafaah

AIP Conf. Proc. 2927, 040016 (2024)
https://doi.org/10.1063/5.0192603

Evaluation of Learning Rate training model on heart disease detection using LSTM

Amrul Faruq[a)], Bellina Rahmamaulida Adeyani[b)], Lailis Syafaah[c)]

Department of Electrical Engineering, Universitas Muhammadiyah Malang, Malang, Indonesia

[a)]Corresponding author: faruq@umm.ac.id
[b)]bellinara@webmail.umm.ac.id
[c)] lailis@umm.ac.id

Abstract. Various algorithms have been used to adjust learning rate parameters, but such strategies generally fail to concentrate on improving the resulting accuracy. Most experts in neural networks use the highest learning rate that allows fusion. The adjustment was made to the weights, and used different adjustment functions to avoid the impact of improper parameter adjustment. In this research, two types of optimizers are used, namely SGD Opt and Adam Opt. In conducting the training and testing process, different learning rate weights are given, namely 0.01, 0.05, and 0.09, with the Adam optimizer, and the use of the default learning rate, namely the SGD optimizer with the learning rate weight obtained automatically 0.000000018. In the conducted experiments, the SGD optimizer with ReduceLRonPlateu gets an average accuracy value of 81% compared to the Adam optimizer, which only gets the highest value of 71% when the learning rate is 0.05, determined manually. It can be concluded that determining the weight value of the learning rate is risky because if the weight of the learning rate is small, then the network takes a relatively long time to occupy or reach a convergence state, even though a small weight will guarantee that the training or testing process will not pass the minimum determination value (0). In contrast, for large weight, the percentage for loss fluctuations when training will be relatively large, so it is difficult to reach a convergence state.

Keywords: LSTM (Long Short Term Memory), Learning rate, Accuracy, Weights

INTRODUCTION

Research using deep learning algorithms has been widely used in identification and classification cases. One that is used in this research is the LSTM (Long Short-Term Memory) algorithm. This algorithm naturally has a learning rate parameter that controls the extent to which the weights can change based on the errors observed and recorded during the training set. Changes in the learning rate can significantly affect the accuracy of the results. Therefore, selecting the learning rate weights for the training algorithm can be problematic, especially when there is no reference value to perform a particular task. Various algorithms have been used to adjust the learning rate parameter, but such strategies generally fail to concentrate on improving the resulting accuracy. Most experts in neural networks use the highest learning rate that allows fusion. However, when the learning rate is set too high, it causes undesirable divergent behavior in the function and leads to losses. Therefore when the highest learning rate is applied to complex and large problems, there is a negative effect on the training process and accuracy.

On the other hand, when the learning rate is set too low, the training progress will be very slow as very small updates are made to the job weights. So there is a need to balance, and there is no better way to do so than by testing multiple learning rates and observing their performance. This work adopts the use of online training instead of batch training. It is because batch training takes more time than online training without the inaccuracy of a suitable upgrade. The LSTM (Long Short Term Memory) algorithm is the most advanced unit of the RNN algorithm because it modifies the RNN by adding memory cells that can store information for a long period [1]. The learning rate in the world of

2nd International Conference on Technology, Informatics, and Engineering
AIP Conf. Proc. 2927, 040016-1–040016-7; https://doi.org/10.1063/5.0192603
Published under an exclusive license by AIP Publishing. 978-0-7354-4901-5/$30.00

deep learning is seen as absolute hyper-parameters that are used to adjust and influence the performance of the training module using the gradient descent algorithm [2, 3]. In the studies conducted various learning rate techniques were obtained, namely inverse square root decay, linear decay, exponential decay, and cosine decay [4, 5]. Learning rates have various solution procedures based on optimization problems. One of the limitations involves the selection in determining the learning rate [6]. Research conducted by Junli Gao, which used the LSTM method for solutions, diagnoses ECG signals more accurately and objectively. It produces an accuracy of 99.26%, recall of 99.26%, the precision of 99.30%, specification of 99.14%, and F1 with scores of 99.27%, respectively [7]. Other studies, which discussed text classification with the combination of CNN and LSTM method, get F1-score results that exceed the other two methods. C-LSTM can obtain an F1-score value of 0.9327 (93.27%) on the selected news dataset, 2.4% better than LSTM and 3.42% better than CNN. In this research, experiments have also been carried out to examine the hyperparameters, namely the learning rate and batch size. However, the learning rate value used only uses the default value of Adam's optimizer, which is 0.001 [8]. The last research related to this research is conducted by Muhammad Bara Al-Farisyi for detecting heart valve function abnormalities with the LSTM method obtained classification results with an average accuracy of 81% and with the ANN method getting classification results with an average accuracy of 75.5% [9]. This study did not heed the value of the learning rate. Therefore, this research will discuss it with an experiment related to changing the value of the learning rate to see and get a higher level of accuracy.

METHODS

In order to answer the problems that have been formulated above, a method is proposed, which is generally described by the block diagram in **FIGURE 1**. The first stage of this research is to search and collect important data, which will be used in the research to be carried out.

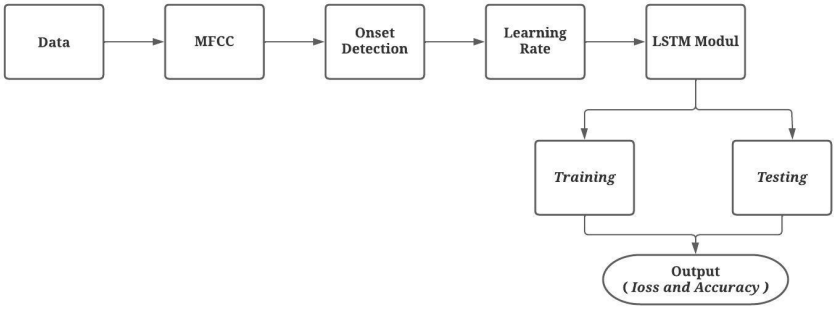


FIGURE 1. Block Diagram

MFCC (Mel Frequency Cepstral Coefficient)

MFCC is one of the audio processing methods that is widely used and recognized in the field of speech technology, especially in the field of sound feature extraction. MFCC works by mapping frequency components with the Mel scale, which will be modeled based on the perception of sound that the human ear can hear. Or it can be said that MFCC performs feature extraction, which will convert the sound signal into several parameters. The formulation of the Mel scale can be seen in **Equation (1)** and **FIGURE 2,** which are the stages in MFCC.

$$M = 2595 \log 10(1\ 700 + 1 \quad (1)$$

Sound Sampling, which takes values and converts them from analog signals into digital signals, is the first step in collecting the heartbeat data. Then in most cases, almost all the energy in sound is stored in the range of 0-4000 Hz, so sampling is sufficient at 8000 Hz. It is based on the Nyquist-Shannon theory, which says that sampling must be at least twice as large as the original signal to prevent loss of information in a continuous to discrete signal conversion.

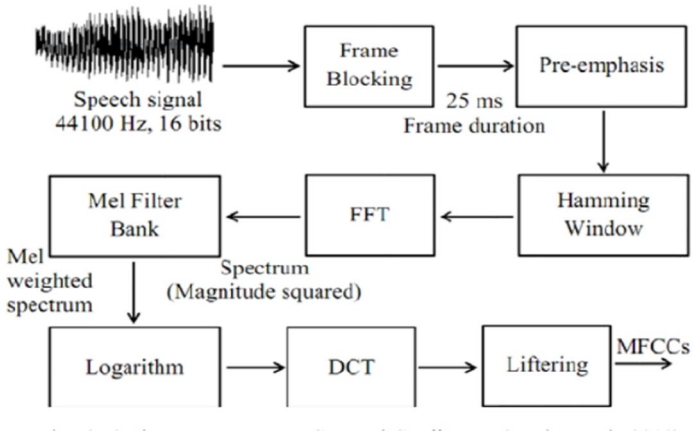


FIGURE 2. MFCC Block Diagram

Frame Blocking, the first process in MFCC calculation where the sound process to be analyzed, is divided into several frames with the same number of sound signals. This process will be carried out continuously until all sound signals can be processed.
Pre-emphasis, carried out to sharpen or clarify the existing sound, is the second stage of MFCC.
Windowing, which aims to minimize discontinuity or leakage at the beginning and the end of the signal, is done on every part of the signal either passed through or generated in the framing process.
Fast Fourier Transform (FFT), the frequency domain, is a method used to analyze signal-level components.
Mel Filter Bank Processing, or MFCC, starts to be calculated by taking a windowed speech signal frame, then using Fast Fourier Transform (FFT) to obtain certain parameters. Then converting to Mel scale to obtain features representing logarithmically compressed amplitude and simple frequency information. Then is calculated by applying the Discrete Cosine Transform (DCT) to the log of the Mel-filter bank. The result is a feature that describes the spectral shape of the signal.
Discrete Cosine Transform (DCT), the final process of Mel-frequency cepstrum coefficients after going through the Mel filter, the Log Mel spectrum needs to be converted into the time domain using Discrete Cosine Transform (DCT). The results will be converted and termed as Mel-Frequency Cepstrum Coefficients.

Onset Detection

Onset detection is one way of processing data that detects the beginning of the sound in this study. Onset detection can be done in several domains, time, frequency, phase, or complex, to find a change in spectral energy at each frequency change. In this research, the onset detection process is carried out in 3 stages, namely:
Onset Detector, the onset detector is based on finding notes at the onset by selecting peaks in the onset strength envelope. The peak_pick parameter is selected by large-scale hyper-parameter optimization over the dataset.
Onset Backtrack, this process can be performed to rewind the detected peak amplitude's timing to the previous minimum. This process is useful when performing onset to determine the slice point for segmentation.

Learning Rate

When training on an artificial neural network, a hyper-parameter with a positive value ranging from 0.0 to 1.0 [10]. This parameter is called the learning rate and can be configured. Learning rate can help in controlling the model in adapting to the given problem. In existing theory, when the weight of the learning rate is smaller, the more the number of epochs because the weight changes that occur will be smaller. Then when the learning rate gets bigger, it results in epochs at high speed. When the learning rate is given a high weight, the model will configure very quickly compared to when given a weight too small. The running process will experience the possibility of stopping or getting stuck in several processes. Therefore, determining the weight of the learning rate is done carefully to get good or maximum results.

LSTM

LSTM is an extension of the RNN method through the addition of LSTM cells in the RNN architecture. LSTM has solved various problems successfully, such as handwriting recognition, speech recognition, handwriting generation, and image captioning [11]. LSTMs allow machine learning architectures to keep the weights of a calculation longer than RNNs. It is because LSTM has LSTM cells, a node that has self-recurrent. It causes LSTM to work better than RNN on data with longer sequences [11]. In LSTM, the recurrent process is performed at the node and layer levels. LSTM cells are controlled by the input gate to remember or forget the information they have based on the output of the LSTM. [12]. In implementing LSTM to perform text classification. The results obtained can also be said to be quite good, with an accuracy value of ≥ 90%. Then, in outline, this research is carried out in stages as described in the block diagram below.

RESULTS AND DISCUSSION

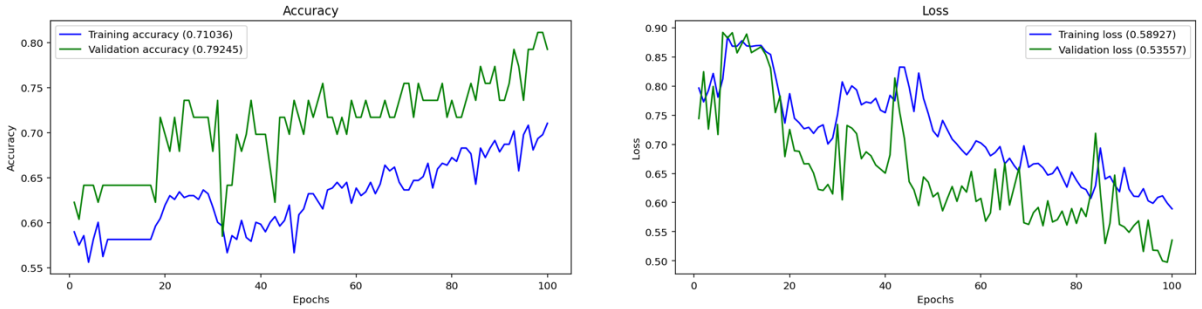
In this data mining, dataset processing is carried out in the first stages because the dataset is still raw data. Therefore, before the data is entered into the main program, processing must be carried out using the Mel Frequency Cepstral Coefficient (MFCC) method and onset detection. Because the form of the dataset is still in the form of sound with wav format, the processing is needed to convert files in the form of voices into files in the form of numbers. Before the results of MFCC are forwarded to the onset detection process, the results will be converted into Discrete Cosine Transform (DCT) first and, at the next stage, will enter the data processing process employing onset detection. Although the data has passed the MFCC and onset detection stages, the data still requires a processing process that includes data merging, filling in blank data, data division, normalization, and data grouping. Therefore data preprocessing is carried out. In this data mining, the data division is divided into 90% train and 10% test by utilizing split data. The intact data will then be used as input to the neural network process. An initial process of the neural network process is to form a model or build a model, where this process will determine the input and output of the algorithm used.

EFFECT OF CHANGING THE LEARNING RATE WEIGHT ON ACCURACY RESULTS

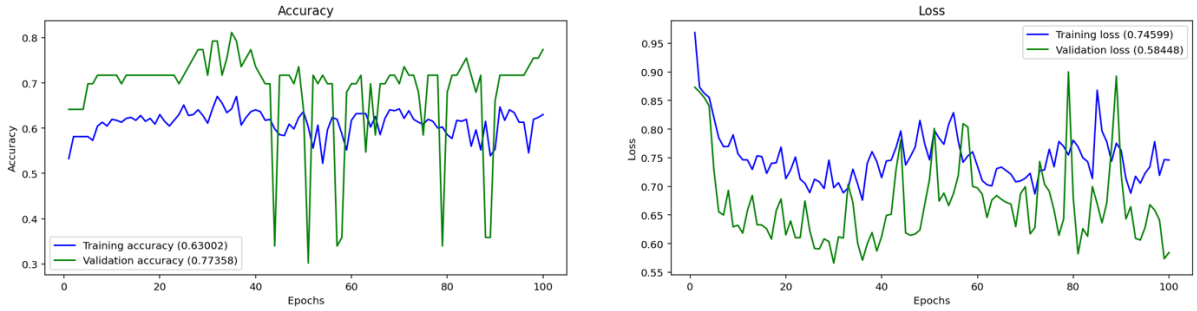
As previously explained, in this data mining, changes are made to the learning rate weight itself. These changes are expected to get the best accuracy value and know the impact of changing the LR on the final accuracy value.

It is easier to know the effect or impact of changing the learning rate weights on the accuracy level in this data mining. This section will discuss the changes that occur in the graph as well as the model accuracy results.

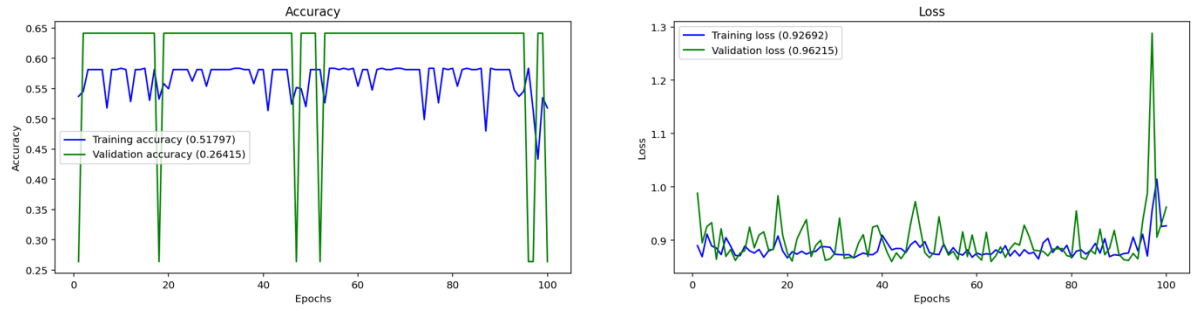
As shown in **TABLE 1,** the learning rates used are 0.01, 0.05, 0.09, and the default LR (automatic) = 0.000000018, which shows a significant change in accuracy. Where LR 0.01 gets a model accuracy of 66%, and 0.05 gets a model accuracy of 71%, 0.09 is 24%, while for the default LR, the largest model accuracy result among other experiments is 81%. Changes that occur are seen not only in the results of the model's final accuracy but also in the accuracy and loss in the training and testing process, shown in **FIGURE 3**.



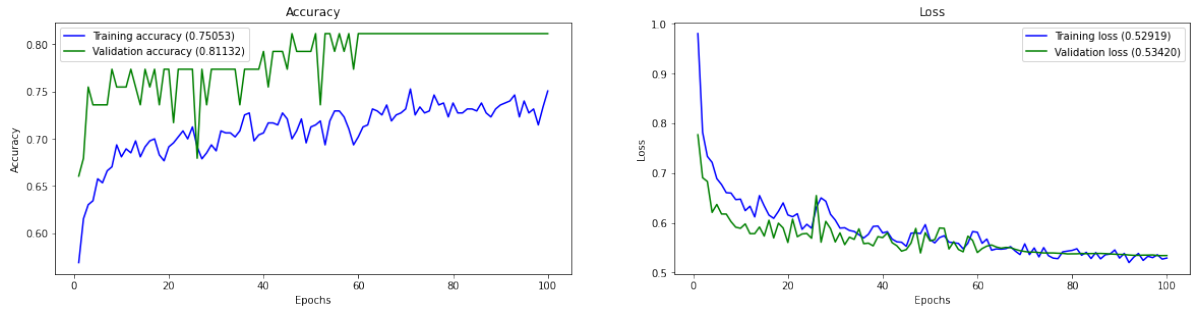
(a)



(b)



(c)



(d)

FIGURE 3. Accuracy and Loss Graph with Learning Rate (a) 0.01 (b) 0.05 (c) 0.09 (d) Default Learning Rate 0.000000018

Figure **3(a)** shows the accuracy and loss graphs on data mining with an LR weight of 0.01, wherein the accuracy graph accuracy reaches the longest stability when the epoch is 85 to 90. In contrast, for loss, it reaches the longest stability when the epoch is in the range of 83 to 85. It can also be seen that the shape of the accuracy training graph (blue) and the accuracy validation graph (green) are not similar and far apart, and the stable period on the validation graph is relatively short. So it can be said that at a learning rate weight of 0.01, the accuracy and loss obtained are not good, so the final accuracy of the model only gets the highest value of 66%. Furthermore, **FIGURE 3(b),** with an LR weight of 0.05, shows that the accuracy reaches twice the longest stability when the epoch is in the range of 5 to 16 and the range of 40 to 60, while the loss shows a dominant unstable state where there are many distractions. The accuracy of the model obtained is 71%.

Similarly, **FIGURE 3(c),** with a learning rate of 0.09, shows many distractions when training, so the graphs that appear are dominantly unstable. To acAchievingility in accuracy occurs when epochs range from 43 to 75, while for the loss, it is difficult to find stability. So, the highest model accuracy obtained is 24%. Finally, **FIGURE 3(d)** shows the training process using the default Learning Rate with a weight of 0.000000018, which is very easy to show the position of stability, where the stability for the longest accuracy is when above the 50th epoch. As for loss, it shows a stable condition when it is in the 50th epoch. And get accuracy results of 81%.

EFFECT OF LEARNING RATE WEIGHT ON SGD OPTIMIZER AND ADAM OPTIMIZER

This section explains the performance of LR against each optimization function used. This research uses adam optimizer for the optimization function when conducting the training and testing. The use of adam optimizer is also shown as a performance comparison with previous research that uses the optimization function, namely SGD optimizer. The accuracy of the program when using the SGD optimizer gets the highest value of 81% when LR is rated 0.000000018 compared to the Adam optimizer, only getting the highest value of 71% when LR is 0.05. Underlying this difference is the selection of the LR weights themselves. When choosing a small LR value, although it can guarantee that it will not miss the minimum determination value (0), it also indicates that the network takes a relatively long time to reach the convergence state, especially when the training is stuck at the saddle point.

In contrast, when LR uses large weights, the loss fluctuation during training will be relatively large, making it difficult to reach the convergence state. In the default learning rate that uses the SGD optimizer, there is a "callback" process where the command operates separately from the optimization algorithm. However, the "callback" work adjusts from the existing LR in the optimization algorithm. Therefore, even though they both use the Adam optimizer, the last experiment was also given the SGD optimizer due to the use of "callback" to use the learning rate schedule. Then there is ReduceLROnPlateau which will adjust the LR when stable model performance is detected or read. The "callback" function reduces LR after the model stops improving to fine-tune the model weights. Also, ReduceLROnPlateau, in theory, requires to specify the metrics to be monitored during the training process through the "monitor" argument, the value or weight of LR to be multiplied through the "factor" argument, and the use of the "patience" argument to specify the number of training epochs to wait before triggering changes to LR. Therefore, even though they both use the adam optimizer, the last experiment also gives the SGD optimizer due to the use of "callback" to use the learning rate schedule.

In contrast, experiments 1 to 3 did not use the "callback" function, and the input of LR weights was only based on the researchers' provisions. So experiments 1 to 3 did not get higher accuracy results. The graph of training and testing process results is as follows in **TABLE 1**.

TABLE 1. Comparison of Accuracy Results Based on Learning Rate Weight

Learning rate	*Model evaluation accuracy*	*Optimizer*
0.01	66%	Adam Optimizer
0.05	71%	Adam Optimizer
0.09	24%	Adam Optimizer
Default learning rate (0.000000018)	81 %	Adam Optimzer and SGD Optimizer

CONCLUSIONS

This research emphasizes changing the learning rate weights in deep learning to determine the impact of these changes and get the best accuracy for a case study of heart disease detection datasets based on heart record sounds. The use of optimizer types also affects the final accuracy level. In conducting the training and testing process, different learning rate weights are given, namely 0.01, 0.05, and 0.09 with the adam's optimizer. Using the default learning rate, namely the SGD optimizer, the LR weight obtained 0.000000018 automatically. In the experiments conducted, the use of the SGD optimizer by utilizing ReduceLROnPlateu gets an average accuracy value of 81% compared to the use of the adam optimizer which only gets the highest value when the LR weight is 0.0,5, and the weight is determined manually. Determining the weight value in LR is a risky thing because if the weight of LR is small, the network takes a relatively long time to occupy or reach a convergence stat,e even though a small weight it will guarantee that the training or testing process will not pass the minimum determination value (0), while for a large weight value, the percentage of loss fluctuations when training will be relatively large, so it is difficult to reach convergence state.

ACKNOWLEDGMENTS

We like to express our thanks to the Department of Electrical Engineering, Faculty of Engineering, University of Muhammadiyah Malang, for the support in helping us finish our research. Without them, this research can not be at the final stage.

REFERENCES

[1] N. K. Manaswi, "Rnn and lstm," in *Deep Learning with Applications Using Python*: Springer, 2018, pp. 115-126.

[2] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*: Springer, 2012, pp. 437-478.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," 2013: PMLR, pp. 343-351.

[5] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701,* 2012.

[6] J. Jepkoech, D. M. Mugo, B. K. Kenduiywo, and E. C. Too, "The effect of adaptive learning rate on the accuracy of neural networks," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 8, 2021.

[7] J. Gao, H. Zhang, P. Lu, and Z. Wang, "An effective LSTM recurrent network to detect arrhythmia on imbalanced ECG dataset," *Journal of healthcare engineering,* vol. 2019, 2019.

[8] Y. Widhiyasana, T. Semiawan, I. G. A. Mudzakir, and M. R. Noor, "Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi,* vol. 10, no. 4, pp. 354-361, 2021.

[9] A. Farisyi and M. Bara, "DETEKSI KELAINAN FUNGSI KATUP JANTUNG DENGAN DEEP LEARNING LONG SHORT TERM MEMORY NETWORK," 2021.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research,* vol. 13, no. 2, 2012.

[11] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning (adaptive computation and machine learning series)," 2016.

[12] J. Li, Y. Xu, and H. Shi, "Bidirectional LSTM with hierarchical attention for text classification," 2019, vol. 1: IEEE, pp. 456-459.