Hindawi
Complexity
Volume 2021, Article ID 5535734, 9 pages
https://doi.org/10.1155/2021/5535734

Research Article

Real-Time System Prediction for Heart Rate Using Deep Learning and Stream Processing Platforms

Abdullah Alharbi,[1] Wael Alosaimi,[1] Radhya Sahal,[2] and Hager Saleh[3]

[1]*Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[2]*Faculty of Computer Science and Engineering, Hodeidah University, Al Hudaydah, Yemen*
[3]*Faculty of Computers and Artificial Intelligence, South Valley University, Hurghada, Egypt*

Correspondence should be addressed to Hager Saleh; hager.saleh.fci@gmail.com

Received 14 January 2021; Revised 27 January 2021; Accepted 10 February 2021; Published 23 February 2021

Academic Editor: Ahmed Mostafa Khalil

Low heart rate causes a risk of death, heart disease, and cardiovascular diseases. Therefore, monitoring the heart rate is critical because of the heart's function to discover its irregularity to detect the health problems early. Rapid technological advancement (e.g., artificial intelligence and stream processing technologies) allows healthcare sectors to consolidate and analyze massive health-based data to discover risks by making more accurate predictions. Therefore, this work proposes a real-time prediction system for heart rate, which helps the medical care providers and patients avoid heart rate risk in real time. The proposed system consists of two phases, namely, an offline phase and an online phase. The offline phase targets developing the model using different forecasting techniques to find the lowest root mean square error. The heart rate time-series dataset is extracted from Medical Information Mart for Intensive Care (MIMIC-II). Recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent units (GRU), and bidirectional long short-term memory (BI-LSTM) are applied to heart rate time series. For the online phase, Apache Kafka and Apache Spark have been used to predict the heart rate in advance based on the best developed model. According to the experimental results, the GRU with three layers has recorded the best performance. Consequently, GRU with three layers has been used to predict heart rate 5 minutes in advance.

1. Introduction

As per the World Health Organization (WHO) [1], heart disease is one of the main death reasons in the world, which reported around 17.7 million deaths worldwide every year. In particular, the heart disease (HD) is caused by any condition affecting the heart when the heart is unable to do its normal function. In particular, the heart cannot push the blood to human body to do the vital functions [2]. In medicine, heart rate (HR) is defined as the number of times the heart beats within a certain time period (i.e., minute), and the normal rate is between 60 and 100 beats per minute. The heart rate data is considered a nonstationary nature, which is unpredictable and cannot be modeled or forecasted. This unpredictability feature increases the risk for the person who has HD and makes him/her suffer coronary disorders. Compared with other risk factors (e.g., age, heredity, hypercholesterolemia, high blood pressure, diabetes, and smoking), HD patient has double the possibility of death risks.

Consequently, technology plays a vital role to early detect the high heart rate to avoid the risk of heart disease progression. Early diagnosis of HD is essential because HD treatment is most effective during the early stages of the disease. To date, many research works have been done using statistical comparative analysis, machine learning, and historical data to estimate the risk factors of diseases. Recently, new technologies like wearable medical sensors have been used to improve medical care by collecting real-time streaming data to be analyzed and deliver impact in

healthcare. The generated time-series streaming data from medical sensors need to be processed on a row-by-row basis by time progression. In particular, the medical time-series streaming data need to be analyzed sequentially and incrementally using streaming sliding time windows concept. This motivates us to integrate time-series data, streaming data, and deep neural network, adding new dimension in healthcare.

1.1. Paper Contribution. In this study, we build a medically oriented real-time prediction system of HR based on conventional deep learning techniques. The predicted HR can help the medical care providers/patients to react quickly to make an instant decision to prevent a crisis in the patient's condition. The contributions of the paper can be summarized as follows:

(1) Developing a time-series streaming data generator to generate HR

(2) Developing a real-time prediction system that helps the medical care providers and patients to avoid the risk of low HR

(3) Finding the best developed model that has the lowest RMSE by applying different deep learning models on historical HR time-series data

(4) Using the best developed model to predict the HR in real time

1.2. Paper Organization. This paper is classified as follows. Related works are described in Section 2. The proposed HR prediction system is described in Section 3. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper.

2. Related Works

Many researchers have used machine learning and deep learning to carry out their works. For example, Masum et al. [3] made a comparison between different deep learning models, LSTM, BI-LSTM, and CNN, using univariate and multivariate time-series data. These models are used to predict blood pressure (BP) 30 minutes in advance and heart rate (HR) 30 minutes in advance as univariate and predict BP and HR as multivariate. Monkaresi et al. [4] proposed machine learning methods to improve the accuracy of HR detection in naturalistic measurements. The results show that the proposed model improved the root mean square error from 43.76 to 3.64 beats/min. Cömert and Kocamaz [5] used recurrent neural networks (RNN), support vector machine, extreme learning machine, radial basis function network, and random forest to classify fetal heart rate signals into normal and hypoxic. The results show that RNN achieved the best performance. Biswas et al. [6] used four-layer deep neural network layers, two CNN layers and two LSTM layers, to predict heart rate. The proposed network was evaluated on the TROIKA dataset having 22 PPG records collected during various physical activities. The result shows that the proposed network improved the mean absolute error for heart rate. Hsu et al. [7] developed a novel deep learning framework to estimate real-time heart rates using an RGB camera. Maragatham and Devi [8] proposed a novel LSTM model for the early diagnosis of heart failure. They compared the proposed models with a support vector machine, logistic regression, k-nearest neighbors, and a multilayer perceptron (MLP). The results show that the proposed model achieved the best accuracy compared to other algorithms.

There are some researches to estimate the heart rate. For example, Bian et al. [9] developed a two-layer LSTM method to estimate pulse signals of heart rate. Shyam et al. [10] proposed a novel end-to-end deep learning model to estimate heart rate using 8-second length input of a Photoplethysmogram (PPG) signal. IEEE Signal Processing Cup (SPC) 2015 database [11] is used to evaluate the proposed model. The proposed model registered 3.36 as a mean absolute error for estimating heart rate. Huang et al. [12] developed deep learning networks that consist of 2D convolutional (Conv2D) and LSTM methods to estimate heart rate.

3. The Real-Time System of Heart Rate Forecasting

Heart rate forecasting system consists of two main phases, an offline phase and an online phase, as shown in Figure 1. The two phases are described in the following.

3.1. Offline Phase. An offline phase targets experimentally finding the best deep learning model that has the smallest RMSE. To do so, five steps are developed:

(1) Data collection

(2) Data preprocessing

(3) Data splitting

(4) Training and optimization models

(5) Evaluation models

Figure 2 describes the main steps of the offline phase, which are described as follows.

3.1.1. Data Collection. In this subsection, we provide a description of the datasets used to train and evaluate the forecasting deep learning models. To do so, an open care dataset called Medical Information Mart for Intensive Care (MIMIC-II) is used [13–16]. We have extracted the heart rate time series (univariate dataset) from MIMIC-II dataset [17] on a minute-by-minute basis for one patient.

3.1.2. Data Preprocessing. In this phase, three steps have been done as follows:

(1) Transforming raw data into stationary data.

Given the nonstationarity of the HR time-series dataset, we have transformed the time-series-based nonstationary dataset into a stationary dataset to be

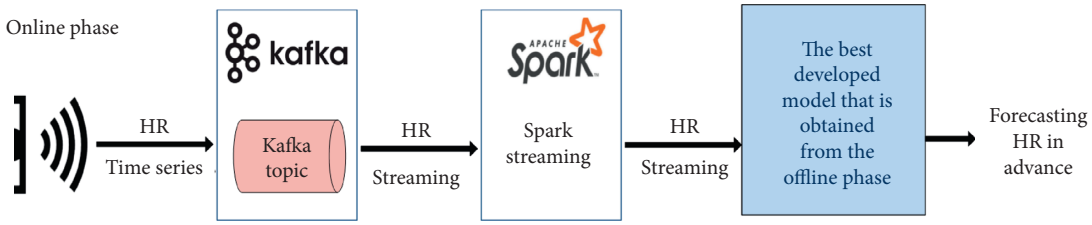

Figure 1: The architecture of HR forecasting system.

fitted in the prediction model. In particular, we have applied a transformation method, which is called differencing. The differencing method's function computes the difference of successive operators in the sequence differencing to escape varying means.

(2) Transforming data into supervised learning.

Within the supervised learning mode, the key idea is that machine learning models learn the association function between input and output variables, denoted by (X) and (Y), respectively. According to the context of this work, which is the real-time prediction, the model has been trained based on input data (X), and then it has been tested to predict the output data (Y) in real time. For converting time-series data to supervised learning, we configurated two windows: observation window (X) and target window (Y). The observation window is configured in 3, 5, and 10 minutes, respectively, while the target window is configured in 5, 7, and 15 minutes, respectively.

(3) Scaling data.

As the work focuses on applying forecasting deep learning models, these models usually work on scaled data within their activation function ranges. We have done a normalization process to scale data between the original range within −1 and a new range within 1. To scale dataset between −1 and 1, we have used the Python library function MinMaxScaler (feature_range = (−1, 1)) [18]. For the evaluation phase, when the model is applied to unseen data, the predictions are transformed back into the original scale. To reverse scaling, the Python library function inverse_transform is used to invert scaling of data row to original values.

3.1.3. Data Splitting. The HR time-series data is split into 80% as a training set and 20% as a testing set. Deep learning models are trained and optimized by the training set and evaluated by the testing set.

3.1.4. Models Optimization and Training. Four deep learning models are used for heart rate forecasting: RNN, LSTM, BI-LSTM, and GRU. According to Figure 3, the deep learning model (i.e., neural network) consists of (1) input sequences in terms of numbers of lags, (2) hidden layers and then dropout layer, and (3) output layer including dense layer that emits its output in three types of forecasting: 5 minutes, 10 minutes, or 15 minutes. The different numbers of hidden layers are applied in hidden layers, including one layer, two layers, and three layers for each model and dropout layer [19]. For the output layer, many neurons have been configured to predict three forecasting times in advance for heart rate (i.e., 5 minutes, 10 minutes, and 15 minutes). In dense layer, rule activation function and Adam optimizer are used [20]. For loss function, mean square error (MSE) is used. For optimization models, a Keras-Tuner library [21] is used to choose the optimal value for two parameters. We configured value range for two parameters: the number of neurons and a dropout rate.

(1) The number of neurons is tuned between 10 and 500

(2) The dropout rate is tuned in a range from 0.1 to 0.9

3.1.5. Evaluating the Models. Root mean square error (RMSE) is utilized to evaluate each model's performance, which is described as follows:

$$\mathrm{RMSE} = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}\left(y_i^{\mathrm{obs}} - y_i^{\mathrm{Pred}}\right)^2}. \quad (1)$$

3.2. Online Phase. In this section, the online phase is introduced. It contains two steps: (1) data generation from the simulated sensor and (2) online prediction. Each step will be described as follows.

3.2.1. Data Generation from the Simulated Sensor. Apache Kafka and Apache Spark are streaming processing platforms that are used to build the online phase. In particular, Apache Kafka [22] has the guarantees of delivery and ordering data streams, which are in demanded the latency-critical applications such as predicting the heart rate in real time. The vital role of Apache Kafka is collecting the health stream data from the wearable sensors (i.e., simulated HR) and then sending it to Apache Spark. Regarding this work, a simulated sensor is developed to generate time-series-based HR dataset in JSON format. The schema of each tuple based on one-minute data consists of two parameters: HR value and timestamp. For the technical level, Kafka Producer API is used to publish the streaming data to the Kafka topic, which is consumed by Apache Spark.

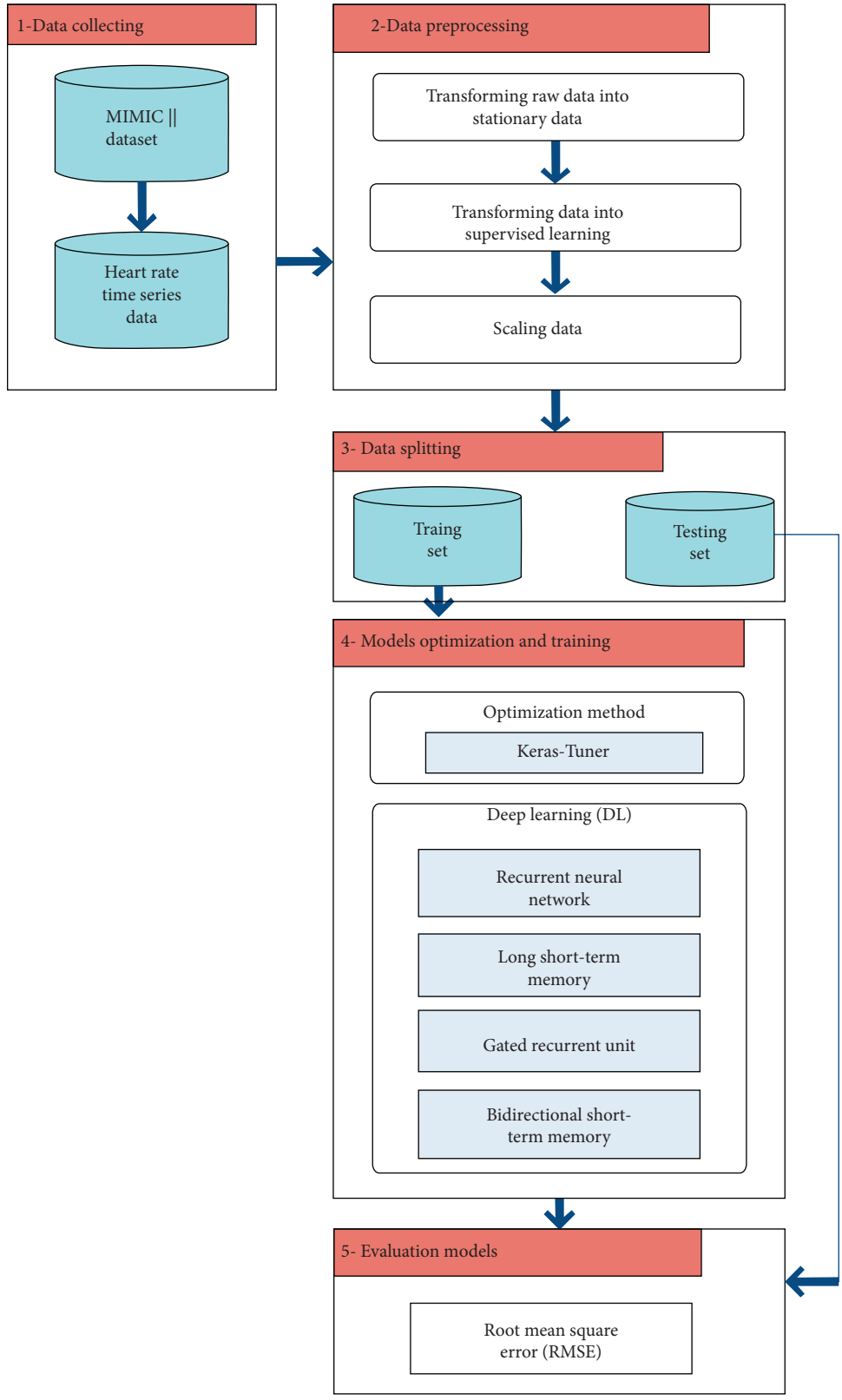

Figure 2: The architecture of the offline phase.

3.2.2. Online Prediction. Apache Spark is a micro-batch-based streaming processing platform. The core principle of Apache Spark is utilizing the memory to analyze streaming data [23]. Furthermore, one of its strengths is ingesting streaming data from different sources including medical sensor. Also, it has a set of streaming APIs, which are able to

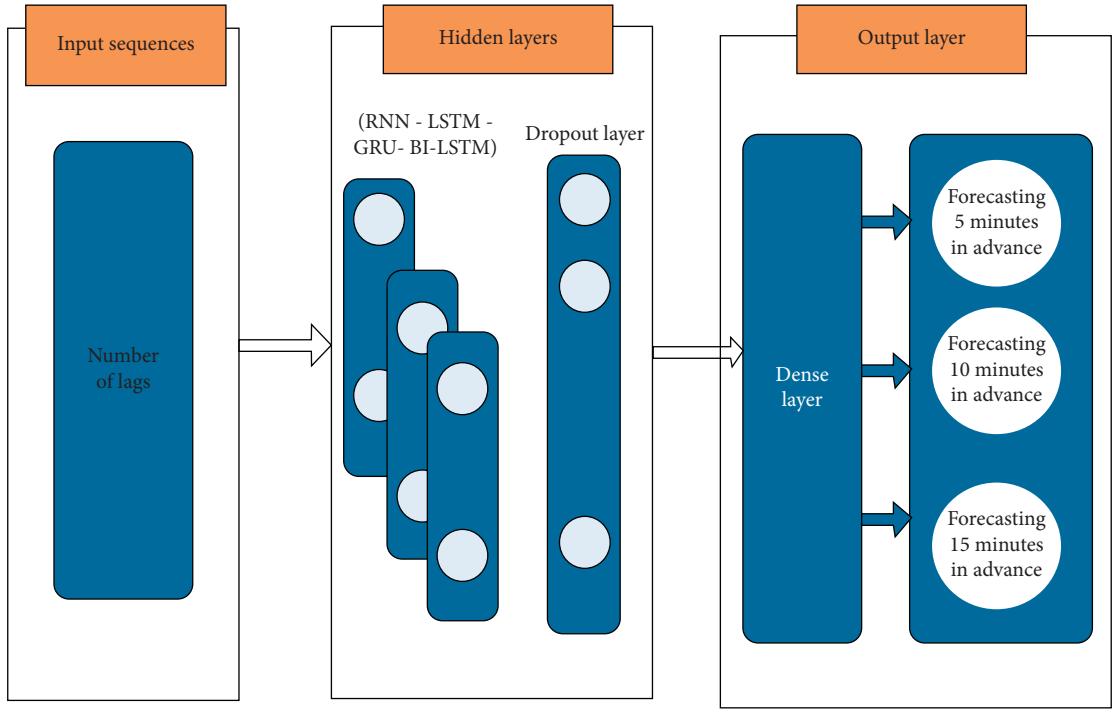

Figure 3: The architecture of neural network for heart rate prediction system.

handle streaming data, and then performing set of complex window-based operations such as map, reduce, and join [24].

According to this work's context, Apache Spark streaming API is used to consume medical data (i.e., HR streams) from precreated Kafka's topic. We have utilized the windowing capability of Apache Spark. Windowing is the heart of any stream processing platform, which splits infinite data stream into chunks of finite data to execute an operation. Streaming window is defined by two parameters: window length and sliding interval. Window length defines window duration, while sliding interval specifies the interval to performing the window operation. The Spark streaming configurations will be described in Experimental Results and Discussion.

4. Experimental Results and Discussion

In this section, the experimental setup, the result of applying deep learning models, and evaluating the proposed system in real time will be described.

4.1. Experimental Setup. Our proposed HR prediction system has been developed using Python libraries, streaming technologies (Apache Kafka and Apache Spark), and deep learning models: RNN, GRU, LSTM, and BI-LSTM. As described earlier, the proposed HR prediction system has been developed in two phases: offline phase and online phase. For the offline phase, deep learning models have been implemented using Python 3.7 and Keras. For the online phase, the selected best developed model has obtained the best performance (i.e., smallest RMSE) used to predict the near future of HR for the next 5 minutes, 10 minutes, or 15 minutes back to 3 minutes, 7 minutes, or 10 minutes of streaming HR data, respectively. For evaluation, we have used RMSE to assess the performance of the deep learning models. Regarding the streaming data generation, we have developed a script that simulates HR sensor data. The developed HR simulated sensor sends the streaming data to the Kafka's topic. To do so, we have developed a Kafka producer using Python, which receives HR streaming data from the simulated sensor and sends it to Kafka's topic. On the other side, Apache Kafka receives the generated HR time-series streaming data and then publishes it to be consumed. Using PySpark, we have created a consumer, mainly Apache Spark, to consume the HR time-series streaming data from Kafka. The online phase has been executed on the Spark cluster for the experimental environment, which comprises one master node and two worker nodes. Ubuntu virtual machines are used to build the cluster. The cluster node characteristics are described in Table 1. Furthermore, we have described further details about the offline phase experimental results, including three lags (i.e., 3, 5, and 10) of minutes using RNN, LSTM, BI-LSTM, and GRU prediction models. Then, we have discussed the online phase using the best developed model obtained by the online phase to predict the risk of HR in the near future.

4.2. The Result of Offline Phase. This section describes the results of performing deep learning models (RNN, LSTM, BI-LSTM, and GRU, using one layer, two layers, and three layers) to the HR time-series dataset. Some values of parameters for each model are configured: lags: 3, 5, and 10 minutes; batch size: 1; learning rate: 0.0001; and epochs: 15.

Table 1: Cluster nodes characteristics.

Parameter	Master	Worker
Processor	Core i7	Core i7
Cores	4	4
Memory	20 GB	20 GB
Disk	100 GB	100 GB
Operating system	Ubuntu 18.04	Ubuntu 18.04

4.2.1. The Result of Deep Learning Models Using Lag of 3 Minutes. We experimentally demonstrate deep learning models' performance using lag of 3 minutes to predict HR in the next 5 minutes in advance (see Table 2). From the RNN model's results, RNN using three layers has obtained the highest performance for all layers (RMSE of 2.370). However, RNN using one layer has obtained the worst performance for all layers (RMSE of 2.459). For the LSTM model, LSTM using three layers has obtained the highest performance for all layers (RMSE of 2.348). However, LSTM using two layers has obtained the worst performance for all layers (RMSE of 2.365). Using two layers for the GRU model has obtained the highest performance for all layers (RMSE of 2.377), and using three layers has done the worst job (RMSE of 2.398). BI-LSTM has recorded the highest performance using one layer, with RMSE of 2.350, while it had the worst performance using three layers, with RMSE of 2.415.

4.2.2. The Result of Deep Learning Models for Lag of 5 Minutes. We experimentally demonstrate deep learning models' performance using lag of 5 minutes to predict HR in the next 7 minutes in advance (see Table 3). From the RNN model's results, RNN using three layers has obtained the highest performance for all layers (RMSE of 2.876). However, RNN using one layer has obtained the worst performance for all layers (RMSE of 2.887). For the LSTM model, LSTM using one layer has obtained the highest performance for all layers (RMSE of 2.822). However, LSTM using three layers has obtained the worst performance for all layers (RMSE of 2.862). Using three layers for the GRU model has obtained the highest performance for all layers (RMSE of 2.817), and using one layer has done the worst job (RMSE of 2.833). BI-LSTM using three layers has recorded the highest performance, with RMSE of 2.810, while it had the worst performance using two layers, with RMSE of 2.931.

4.2.3. The Result of Deep Learning Models for Lag of 10 Minutes. We experimentally demonstrate deep learning models' performance using lag of 10 minutes to predict HR in the next 15 minutes in advance (see Table 2). From the RNN model's results, RNN using two layers has obtained the highest performance for all layers (RMSE of 3.100). However, RNN using one layer has obtained the worst performance for all layers (RMSE of 3.113). For the LSTM model, LSTM using three layers has obtained the highest performance for all layers (RMSE of 3.022). However, LSTM using one layer has obtained the worst performance for all layers (RMSE of 3.043). Using three layers for the GRU model has obtained the highest performance for all layers (RMSE of 3.043), and using two layers has done the worst job (RMSE of 3.074). BI-LSTM has achieved the highest performance using one layer, with RMSE of 3.052, while it had the worst performance using two layers, with RMSE of 3.263.

4.3. Discussion. In comparison of the four deep learning models for lag of 5 minutes, it is noted that the BI-LSTM model using three layers has recorded the highest performance with respect to other models (RMSE of 2.810). However, the lowest performance has been obtained by BI-LSTM using two layers, with an RMSE of 2.931. Similar to lag of 5 minutes using the four deep learning models for one layer, two layers, and three layers, it can be seen that the BI-LSTM model using three layers has recorded the highest performance among all models (RMSE of 2.810). However, the lowest performance has been obtained by BI-LSTM using two layers, with an RMSE of 2.931. For lag of 10 minutes, it is noticed that the LSTM model using three layers has recorded the highest performance with regard to other models (RMSE of 3.022). However, the lowest performance has been obtained by BI-LSTM using two layers, with an RMSE of 3.263.

Given the empirical results seen in Tables 2-4, Figure 4 depicts the best model that has recorded the four models' highest performances using lags of 3, 7, and 10 minutes in the offline phase of the proposed HR prediction system. It can be noted that the best performance is obtained by the GRU model using three layers for lag of 10 minutes to forecast HR 15 minutes in advance. Given the consideration of the findings based on our experimental results, GRU model using three hidden layers has been chosen to predict the near future of the HR (i.e., 15 minutes in real time).

4.4. Evaluation of the Proposed System in Real Time. For online phase, we have used the best developed model that has recorded the best performance among all deep learning models to evaluate the HR system in real time. According to the aforementioned results, we have chosen LSTM model using three hidden layers for lag of 3 minutes to predict the HR in real time 5 minutes in advance using streaming HR time-series data. The HR simulated sensor that sends the streaming HR time series is used to evaluate the ability of our proposed system to work in real time. The capability of predicting the HR rates is beneficial to warn the patients about the potential risks to take the proper actions.

As mentioned earlier, HR streaming time-series data is collected using simulated HR sensor data. Apache Kafka receives the collected streaming data and publishes it. Then, Apache Spark streaming consumes the published HR streaming data. Based on the prediction of streaming data using lag of 3 minutes, Spark sliding window parameters have been configured as X and Y for window length and sliding interval, respectively. The preprocessing steps are applied including data scaling, data transformation, and data inversion. Afterwards, the predicted HR rate based on the streaming HR time-series data is used to warn the patient 5 minutes in advance in case of HR risk.

Table 2: The result of DL models for lag of 3 minutes and forecasting 5 minutes in advance.

DL models	No. of layers	No. of neurons	Dropout	RMSE
RNN	1	120	0.3	2.459
	2	[60,120]	[0.5,0.1]	2.387
	3	[60,80,40]	[0.4,0.2,0.4]	2.370
LSTM	1	50	0.5	2.355
	2	[100,140]	[0.4,0.2]	2.365
	3	[180,120,20]	[0.3,0.3,0.3]	2.348
GRU	1	150	0.2	2.38
	2	[110,30]	[0.3,0.3]	2.377
	3	[60,150,70]	[0.4,0.3,0.3]	2.398
BI-LSTM	1	90	0.2	2.350
	2	[60,140]	[0.4,0.3]	2.377
	3	[100,120,100]	[0.3,0.2,0.4]	2.415

Table 3: The result of DL models for lag of 5 minutes and forecasting 7 minutes in advance.

DL models	No. of layers	No. of neurons	Dropout	RMSE
RNN	1	100	0.4	2.887
	2	[200,40]	[0.5,0.1]	2.878
	3	[100,140,60	[0.5,0.4,0.4]	2.876
LSTM	1	100	0.4	2.822
	2	[50,130]	[0.5,0.1]	2.825
	3	[170,50,70]	[0.4,0.4,0.4]	2.862
GRU	1	50	0.3	2.833
	2	[200,140]	[0.3,0.3]	2.831
	3	[140,50,180]	[0.3,0.3,0.3]	2.817
BI-LSTM	1	80	0.4	2.823
	2	[220,80]	[0.5,0.4]	2.931
	3	[120,100,180]	[0.2,0.3,0.3]	2.810

Table 4: The result of DL models for lag of 10 minutes and forecasting 15 minutes in advance.

DL models	No. of layers	No. of neurons	Dropout	RMSE
RNN	1	160	0.4	3.113
	2	[100,80]	[0.5,0.1]	3.100
	3	[100,120,40]	[0.2,0.5,0.4]	3.104
LSTM	1	140	0.3	3.043
	2	[70,40]	[0.2,0.2]	3.031
	3	[130,150,110]	[0.4,0.2,0.4]	3.022
GRU	1	110	0.4	3.068
	2	[110,190]	[0.3,0.4]	3.074
	3	[160,60,70]	[0.3,0.4,0.2]	3.043
BI-LSTM	1	40	0.5	3.052
	2	[60,80]	[0.5,0.3]	3.263
	3	[160,60,60]	[0.4,0.4,0.3]	3.0582

Table 5 describes generated data samples and their corresponding predicted HR. The developed simulation-based HR sensor sends the generated HR reading on a minute-by-minute basis to be ingested into Kafka topic. The six ingested HR readings for the first 3 minutes are 56.8616, 57.0785, and 56.898, which are consumed by Spark every X minutes with 1-minute slide. Then, these consumed streaming HR time-series data are sent to the online model which applied the best prediction model (i.e., GRU using three hidden layers) to predict HR 15 minutes in advance in real time.

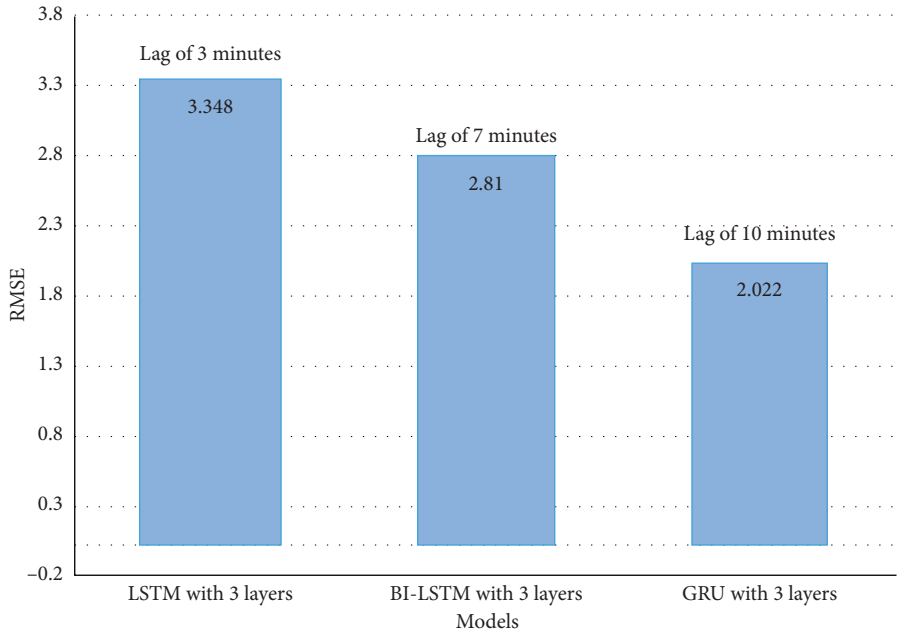

Figure 4: RMSE-based comparison of best prediction models.

Table 5: Generated data samples and their corresponding predicted HR.

Time (min)	HR in Kafka queue	Spark streams	Predicted HR
1–3	[56.8616, 57.0785, 56.898]	[56.8616, 57.0785, 56.8985]	[60.3684, 61.4218, 62.7123, 63.9552, 65.3004]
1–4	[56.86163, 57.0785, 56.8985, 59.7279]	[57.0785, 56.8985, 59.7279]	[59.8236, 60.8004, 61.7510, 62.7102, 63.6634]
3–5	[56.8616, 57.0785, 56.8985, 59.727, 58.8089, 58.5289]	[56.8985, 59.7279, 58.8089]	[59.5885, 60.5639, 61.5514, 62.4929, 63.4583]

5. Conclusion

This paper has addressed the prediction of HR in real time by introducing an HR prediction system to detect the early risk of low HR. The proposed real-time prediction system consists of two phases: the offline phase and the online phase. In the offline phase, RNN, LSTM, GRU, and BI-LSTM using one layer, two layers, and three hidden layers are used to train and evaluate HR time-series data. In addition, three cases of forecasting HR in advance were made: 5 minutes with lag of 3 minutes, 10 minutes with lag of 7 minutes, and 15 minutes with lag of 10 minutes. The best developed model for each case that has the lowest RMSE is used to evaluate the system in real time. In the online phase, the developed simulation-based sensor generates HR time-series data and sends them to the Kafka Topic. Then, Spark streaming reads HR data from the Kafka topic and sends them to the best developed model to predict HR in the near future 15 minutes in advance. Regarding the experimental evaluation, the GRU model has recorded the best performance using three hidden layers. Therefore, we have chosen it to be used to predict the HR hear rate in real time (i.e., 5 minutes). The beneficiaries of the proposed real-time prediction system are the medical care providers and patients to utilize the real-time streaming data processing and deep learning.

Data Availability

The heart rate time-series data are extracted from MIMIC-II dataset (https://archive.physionet.org/cgi-bin/atm/ATM).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by Taif University Researchers Supporting Project number (TURSP-2020/231), Taif University, Taif, Saudi Arabia.

References

[1] World Health Organization, *Heart Disease*, World Health Organization, Geneva, Switzerland, 2020, https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1.

[2] A. L. Bui, T. B. Horwich, and G. C. Fonarow, "Epidemiology and risk profile of heart failure," *Nature Reviews Cardiology*, vol. 8, no. 1, pp. 30–41, 2011.

[3] S. Masum, J. P. Chiverton, Y. Liu, and B. Vuksanovic, "Investigation of machine learning techniques in forecasting of blood pressure time series data," in *Proceedings of the International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 269–282, Springer, Cambridge, UK, December 2019.

[4] H. Monkaresi, R. A. Calvo, and H. Yan, "A machine learning approach to improve contactless heart rate monitoring using a webcam," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1153–1160, 2013.

[5] Z. Cömert and A. F. Kocamaz, "Comparison of machine learning techniques for fetal heart rate classification," *Acta Physica Polonica A*, vol. 132, no. 3, pp. 451–454, 2017.

[6] D. Biswas, L. Everson, M. Liu et al., "CorNET: deep learning framework for PPG-based heart rate estimation and biometric identification in ambulant environment," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 2, pp. 282–291, 2019.

[7] G.-S. J. Hsu, R.-C. Xie, A. Ambikapathi, and K.-J. Chou, "A deep learning framework for heart rate estimation from facial videos," *Neurocomputing*, vol. 417, pp. 155–166, 2020.

[8] G. Maragatham and S. Devi, "LSTM model for prediction of heart failure in big data," *Journal of Medical Systems*, vol. 43, no. 5, p. 111, 2019.

[9] M. Bian, B. Peng, W. Wang, and J. Dong, "An accurate LSTM based video heart rate estimation method," in *Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 409–417, Springer, Xi'an, China, November 2019.

[10] A. Shyam, V. Ravichandran, S. Preejith, J. Joseph, and M. Sivaprakasam, "PPGnet: deep network for device independent heart rate estimation from photoplethysmogram," in *Proceedings of the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1899–1902, IEEE, Berlin, Germany, July 2019.

[11] Z. Zhang, "Spc," http://h://sites.google.com/site/researchbyzhang/ieeespcup2015, 2020.

[12] B. Huang, C.-M. Chang, C.-L. Lin, W. Chen, C.-F. Juang, and X. Wu, "Visual heart rate estimation from facial video based on CNN," in *Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1658–1662, IEEE, Kristiansand, Norway, November 2020.

[13] M. Saeed, C. Lieu, G. Raber, and R. G. Mark, "MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring," *Computers in Cardiology*, vol. 29, pp. 641–644, 2002.

[14] J. Sun, A. Reisner, M. Saeed, and R. Mark, "Estimating cardiac output from arterial blood pressure wave forms: a critical evaluation using the MIMIC II database," in *Proceedings of the Computers in Cardiology, 2005*, pp. 295–298, IEEE, Lyon, France, September 2005.

[15] A. Mikhno and C. M. Ennett, "Prediction of extubation failure for neonates with respiratory distress syndrome using the MIMIC-II clinical database," in *Proceedings of the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5094–5097, IEEE, San Diego, CA, USA, August 2012.

[16] D. J. Scott, J. Lee, I. Silva et al., "Accessing the public MIMIC-II intensive care relational database for clinical research," *BMC Medical Informatics and Decision Making*, vol. 13, no. 1, p. 9, 2013.

[17] ——, "MIMIC II," https://archive.physionet.org/cgi-bin/atm/ATM, 2021.

[18] sklearn, "Minmaxscaler," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html, 2020.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, http://arxiv.org/abs/1207.0580.

[20] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[21] kerastuner, "kerastuner," https://keras-team.github.io/keras-tuner/, 2021.

[22] A. Kafka, "Apache Kafka," https://kafka.apache.org/, 2021.

[23] A. Spark, "Apache Spark," https://spark.apache.org/, 2021.

[24] A. Spark, "Spark Streaming," https://spark.apache.org/docs/latest/streaming-programming-guide.html, 2021.