Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8773900, 7 pages
https://doi.org/10.1155/2022/8773900

Research Article

A Multihead ConvLSTM for Time Series Classification in eHealth Industry 4.0

Yilin Wang[1], **Le Sun**[1] and **Dandan Peng**[2]

[1]*Engineering Research Center of Digital Forensics, Ministry of Education, Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET) Nanjing University of Information Science & Technology, Nanjing 210044, China*
[2]*School of Computer Science and Network Engineering, Guangzhou University, Guangdong, China*

Correspondence should be addressed to Le Sun; 002813@nuist.edu.cn

Received 21 January 2022; Accepted 8 February 2022; Published 1 March 2022

Academic Editor: Kalidoss Rajakani

Healthcare time series classification is to classify the collected human physiological information based on artificial intelligence technologies. The main purpose is to use pattern recognition technology to enable machines to analyze characteristics of human physiological signals based on deep learning in electronic health (E-health) industry 4.0. Healthcare time series classification can analyze various physiological information of the human body, make correct disease treatments, and reduce medical costs. In this paper, we propose a multiple-head convolutional LSTM (MCL) model for healthcare time series classification. MCL is a convolutional LSTM (ConvLSTM) model with multiple heads. It can extract both time and spatial features of healthcare data and increase the number of features to achieve more accurate classification results.

1. Introduction

In recent years, the application of pattern recognition technology in healthcare data analysis and processing has become more and more extensive. The intelligent classification of healthcare data greatly improves the capability and efficiency of medical services. In Industry 4.0, the healthcare has also made big progress. The field of medicine is one of the main applications of artificial intelligence (AI). We can speculate on the probability of people getting sick based on known information, trace the source of the disease through big data, and inform patients to go to the hospital for detailed treatment.

Much work has been done to analyze the healthcare time series based on AI technology. Irfan and Hameed proposed implementing deep summary neural networks (DCNN) to solve the problem of classifying image pattern data extracted from electrocardiograms (ECGs) [1]. Hartmann et al. [3] combined 5G and AI and used machine learning to apply medical applications. This work promotes the application of AI deep learning in the field of medicine and effectively promotes the development of modern medicine.

Through these researches, we find that the improvement of the accuracy of Healthcare Timing Classification has always been a focus and difficulty in academic research. Fortunately, a great deal of work has been done on this issue, such as Irfan's deep learning for healthcare data system [1]. Guo et al. [9] used deep learning LSTM and 5× crossvalidation models to predict heart disease.

One of the main problems that makes Healthcare Timing Classification accuracy difficult to improve is category imbalance. Category imbalance means that the number of training samples for different categories varies too much. There is a widespread imbalance in medical data. We need to target it. TensorFlow Core suggests that category imbalances can be addressed with data enhancement and standardization. This paper also puts forward a method to solve the category imbalance, which includes normalizing the data, trimming the normalized data, and adjusting the weight of the sample label [7]. In addition, we have used

L2 regularization in training to address overfitting issues caused by category imbalance.

In order to improve the accuracy of Healthcare Timing Classification, an MCL model is proposed to implement Healthcare Timing Classification.

The main purposes of this article are as follows:

(1) Propose a method of normalization, pruning, and adjusting the weight of sample labels to solve the problem of category imbalance

(2) Design a MCL model for high-accuracy healthcare timing classification

(3) Conduct several experiments based on HAR, MIT-BIH, and Sleep-EDF to verify the effectiveness of MCL

The structure of this article is as follows: Section 2 introduces the relevant work. Section 3 describes the proposed multiple-head convolutional LSTM (MCL) model structure. Section 4 introduces the experimental process and results. Section 5 summarizes the article and puts forward the future research direction.

2. Related Work

Bar et al. [14] used CNN to train neural networks for chest X-ray images to classify different types of pathology. Rezaei et al. [15] used deep learning for end-to-end training of medical images. Irfan and Hameed [1] proposed a deep summary neural network (DCNN) to classify electrocardiograms. Kermany et al. [13] used transfer learning to conduct training analysis of scanned images for medical diagnosis and treatment of diseases. Residual networks and end-to-end learning methods have been widely used. Bian et al. [18] used8 PPG signals to estimate respiratory rate (RP). Kisa et al. [2] proposed the 101-layer converged neural network based on the residual network (ResNet) to identify movements and gestures. Guo et al. [9] demonstrated the advantages of LSTM and predicted heart disease through 5× crossvalidation. Sridhar et al. [11] used deep learning models to analyze instantaneous heart rate and classify the sleep phase. Ali et al. [10] used a deep, aggregated neural network architecture to identify human activity data sets. Alharbi et al. [16] used RNN, LSTM, GRU, and BL_LSTM techniques to build models to compare heart rate time series. Suryani et al. [17] used musk R-CNN to construct a model to automatically detect objects in X-ray images to improve the accuracy of radiation detection.

Data imbalance is a common problem, which can be addressed by data enhancement and standardization [7]. Goschenhofer et al. [12] used a grade-weighted approach to balance data for motion status classification. Shaker et al. used the Build-Adversarial Network to balance the data and proposed a deep summary neural network hierarchical approach to classify signals. Romdhane et al. [4] used focus loss function to address data imbalances problem.

In this article, a multiple-head convolutional LSTM (MCL) model is proposed, which combines the characteristics of CNN and LSTM. The problem of information loss encountered in LSTM processing is effectively solved. And in a Multi_head way, the model can operate simultaneously using convolution cores of different sizes, which can provide better performance. In addition, we have tried to solve the problem of data imbalance by using normalization processing and trimming data. This method effectively increases the accuracy of the MCL model. In this article, MCL performance is demonstrated by using three datasets: human activity recognition (HAR), MIT-BIH, and Sleep-EDF. And we compare the performance of MCL with several deep learning neural networks.

3. Multiple-Head Convolutional LSTM

The multiple-head convolutional LSTM (MCL) model is ConvLSTM with multiple_heads model, and each One_head ConvLSTM of the model can read input time steps using convolution cores of different sizes. For example, in lab 1, we used the MCL model of Three_head, and for three different One_head ConvLSTM, we used three different convolution cores (1, 3), (1, 5), and (1, 11). This allows the model to read and interpret input data in three different filter sizes [6]. The three One_head then runs the results of ConvLSTM in the merge layer (merge) connection. The full-connection layer (FC) performs the prediction operation.

The MCL architecture uses convolutional neural networks (convolutional LSTM) in conjunction with LSTM to process input data, perform functional extraction to support sequence prediction, and further expansion, and MCL will CNN Summarized as part of LSTM. Unlike LSTM, which reads data directly to calculate internal state and state transitions, ConvLSTM uses convolutions directly as part of the LSTM's own input [5].

3.1. The Proposed Multiple-Head Convolutional LSTM. The initial data of the experiment had problems with the imbalance of sample classification, and we normalized the data and trimmed the normalized data (for example, set the data greater than 5 to 5 and the value less than -5 to -5), making it easy for us to perform experimental calculations. Normalization processing demeans and scales variances for each feature dimension, so that the processed data conforms to the standard normal distribution. We divide the data into training sets and test sets. The training set is normalized and trimmed, and the test set is normalized and trimmed using the mean and variance obtained by the training set, so as not to interfere with the test set and make false experimental results. The formula for normalization processing is formula (1).

$$X^* = \frac{X - \mu}{\sigma}, \tag{1}$$

where X is the initial input, X^* is the data after normalization, μ is the mean of the training set sample data, and σ is the standard deviation of the training set sample data.

Data that is balanced is a three-dimensional data $X = [x_1, x_2, \cdots x_n]^T$ shaped (sampleNum, timesteps, channel).

The first sample, the second sample, and the nth sample are the first, and each sample has a shape of $x_1, x_2, \cdots x_n$ (1, timesteps, channel), and sampleNum is the sample number. Timesteps are the time series for each sample, and channel is the number of channels that the sample data has. In our proposed MCL model, ConvLSTM has convolution structures from incoming to state and state-to-state transitions, which requires us to expand the dimensions. For example, we took ConvLSTM2D as the ConvLSTM of MCL in the processing of a pair of the Human Activity Recognition (HAR) Dataset, and then we need to reshape 3D data (sampleNum, timesteps, channel) into 5D data (sampleNum, time, row, cols, channel). In MCL, we keep the sample and channel of the input data and reshape of the timesteps 1D data for 3D data (time, row, col), time is timing input, and row and col are rows and columns, respectively [5]. When we process data, we make timesteps of all samples the same shape to avoid losing chronological order during the reshape process. The data preprocessing steps are shown in Figure 1.

3.2. The Framework of MCL

3.2.1. ConvLSTM. CNN uses convolutional operations to extract spatial features, LSTM uses memory units and doors to extract time characteristics, and ConvLSTM combines the characteristics of CNN and LSTM to take advantage of both space-time characteristics. The heart of ConvLSTM is convolution of memory units and doors. C_t ConvLSTM acts as an accumulator for status information. The door is controlled by adaptive training parameters. Whenever there is a new input, the information that the input door i_t controls it accumulates in the memory unit C_{t-1}. Forget the door f_t control the cell state of the past C_{t-1} will it be "forgotten" during this procession. The output gate o_t controls whether the final unit output will be entered to the final state [8]. In general LSTM, it (f_t, o_t, i_t) is a 1D vector that loses information when processing space-time data. ConvLSTM is a three-dimensional vector [8]. With convolution operations, ConvLSTM not only gets time-to-time features but also extracts spatial features like convolution layers. Data collected by devices in the Healthcare timing classification in time intervals has a single time series with no spatial properties. We change the shape of the experimental data by preprocessing, giving the data spatial properties. Time and spatial features are extracted at the ConvLSTM layer of the MCL model, after which both time and spatial features are interpreted and classified at the full connection layer. Figure 2 shows the internal structure of ConvLSTM. H is a hidden state, and a cell state, $C_t\chi_t$, is input data.

Formulas (2) to (6) are the key formulas of ConvLSTM [8]. H is a hidden state, W is a filter, i_t indicates an input door, χ_t is the input,f_t indicates a forgotten door, C_t is a cell state, o_t is an output door, and b is a bias. And "$*$" means convolution operator, and "$\circ$" represents Hadamard product [8].

$$i_t = \sigma(W_i * \chi_t + W_i * H_{t-1} + W_{ci} \circ C_{t-1} + bi), \quad (2)$$

$$f_t = \sigma\left(W_f * \chi_t + W_f * H_{t-1} + W_{cf} \circ C_{t-1} + bf\right), \quad (3)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh\left(W_c * X_t + W_c * H_{t-1} + b_c\right), \quad (4)$$

$$o_t = \sigma(W_o * \chi_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + bf), \quad (5)$$

$$H_t = o_t \circ \tanh\left(C_t\right). \quad (6)$$

3.2.2. One_head ConvLSTM. A standard One_head ConvLSTM model consists of four layers: input, ConvLSTM, dropout, and flatten (in order). We first enter the data input layer, which preprocesses the data, and records our data for next training. The ConvLSTM layer is the core layer of our model, which determines the future state of a cell in the grid by its input and past state, and extracts spatial and temporal features through its convolution core and cell state, gates. We use the correction linear unit $f(x) = \max(0, x)$ (ReLU) as the ConvLSTM layer activation function. The dropout layer is randomly inactivated, and we take a random inactivated 50% in this layer to reduce overfitting. The final flatten layer "flattens" the input data, which is the one-dimensional input of the multidimensional layer for the transition from the convolution layer to the full connection layer. To reduce the degree of overfitting, we regularize L2 for each layer that has data processing. Formula (7) is the L2 regularization function.

$$L^* = L + \lambda \sum_{i=1}^{n} W_i^2, \quad (7)$$

where L^* is the regularized loss function, L is the original loss function, W_i is the weight, n is the number of weights, and λ is a regularized parameter.

3.2.3. MCL. In our experiment, we chose Three_head to build the MCL model. Three_head that the actual effect of the model is not too expensive. MCL uses One_head ConvLSTM of different convolution core sizes to extract different space-time features at the same time and then connects space-time features extracted by different convolution cores at the merge layer. This equates to a sample with more space-time characteristics and more accurate predictions. Therefore, generally speaking, the more head, the higher the accuracy of the model. In an MCL model, each One_head ConvLSTM model can select different superparameters, such as the number of filters, the size of the filters, activation functions, and L2 regularization parameters. The merged layer then passes the resulting data to the full-connected layer (FC). The full connection layer is processed once, fused with the space-time characteristics learned, and then classified by $S_i = e^i / \sum_j e^j$ (softmax). The loss function is a multiclassified loss function (categorical_crossentropy). Figure 3 shows the architecture used by the MCL model in activity recognition (HAR). Formula (8) shows the multiclassification loss function (categorical_crossentropy),

$$\text{Loss} = -\sum_{i=1}^{\text{size}} y_i * \log y_i^{\text{pred}}, \quad (8)$$

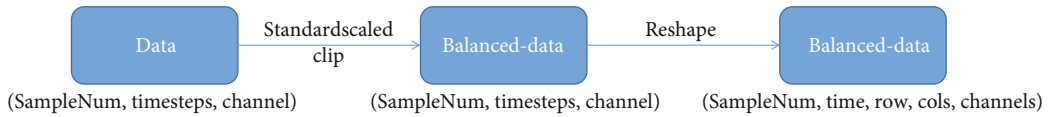

Figure 1: Data preprocessing.

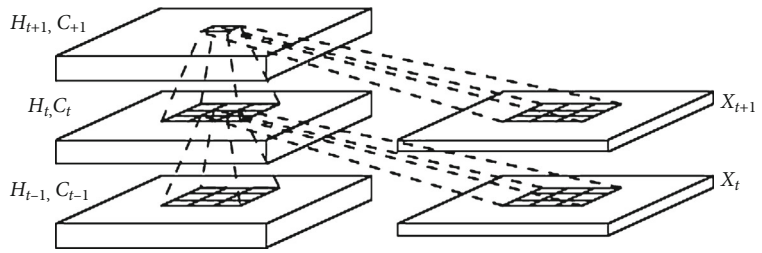

Figure 2: Inner structure [8].

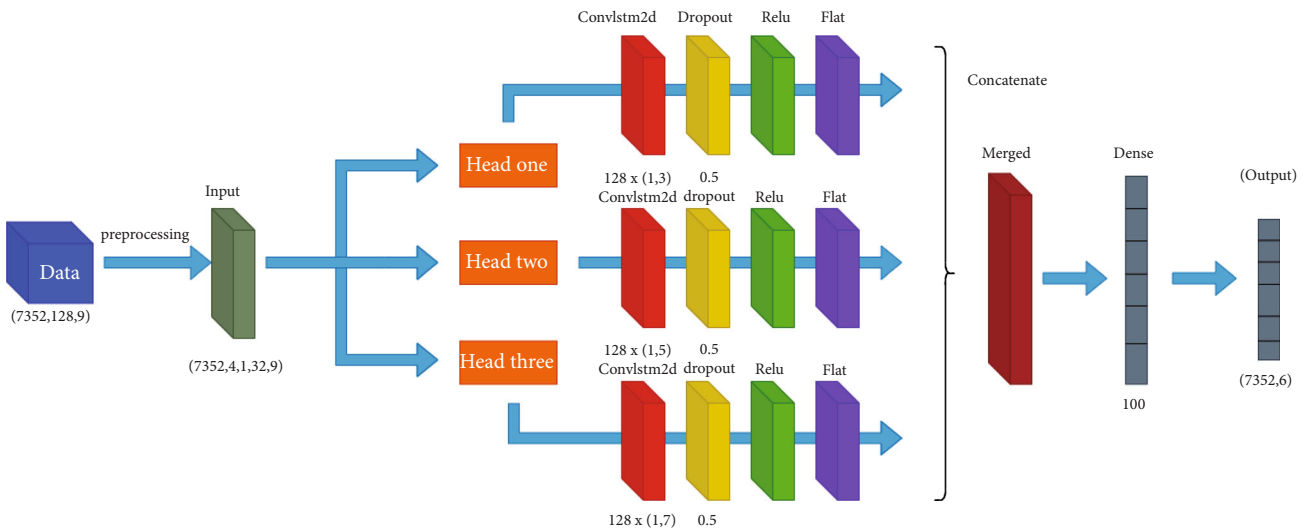

Figure 3: MCL on HAR.

where y^{pred} is the predicted quantity, and y is the actual quantity.

3.3. Model Evaluation. Due to the imbalance in the number of sample classifications, in addition to the correct rate, we need more indicators to evaluate the model, so as not to fall into the trap of high accuracy and low practicality. We chose confusion matrix to visualize the performance of the model, and in confusion matrix, we used accuracy, precision, and recall, F1_score to accurately reflect the performance of the model. Table 1 shows the confusion matrix, and formulas (9) to (12) show the precision, recall, and F1_score.

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{TN}}, \tag{9}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{12}$$

where FN, TN, FP, and TP denote false negatives, true negatives, false positives, and true positives, respectively.

3.4. Superparameter Setting. The number of filters (filters) is an important superparameter of ConvLSTM, and in order to extract enough features, the convolution layer should start wide enough. The more complex the problem, the wider the convolution layer, that is, the larger the number of filters. However, sometimes in order to reduce overfitting problems, you can choose a small number of filters for training. The number of filters can be adjusted dynamically. Select the most appropriate quantity after multiple comparisons. The number of filters is generally increased and decreased by multiples. In training, we select 16 and 32 as the number of filters.

The size of the filter (kernel_size) is another important superparameter of ConvLSTM, where the kernel size controls the number of time steps considered in each read in the input sequence and then projected onto the function

Table 1: Confusion matrix [19].

Confusion matrix		Pred	
		0	1
True	0	TN	FP
	1	FN	TP

diagram (via a summary process). Large size means that the reading of data is less rigorous, but generally has a faster training speed. We used (1, 3), (1, 5), (1, 7), and so on as filter sizes in our experiments.

Batch size is the number of samples selected for a training session. The size of batch size affects how optimized and faster the model is. It also has a direct impact on GPU usage, and in general, choosing a larger batch size can reduce overproposed merges and speed up training. After many comparative experiments, we finally chose 16 to train.

The number of iterations (epochs) is the number of times training is optimized for the same batch of data. Theoretically, the more training you have in general, the smaller the loss function and the higher the accuracy. But generally, batch size is larger, and to reach the same accuracy, you have to increase the epoch, but too high epochs can cause serious overfitting. We chose 32 to train.

L2 regularization can be seen as a penalty for the loss function. Its parameters can be adjusted to reduce overfitting. We used 0.005 as the L2 parameter in our experiment.

4. Experiments and Results

To evaluate the MCL model, we used three data sets: human activity recognition (HAR), MIT-BIH, and Sleep-EDF. The experimental equipment we use during training is a personal notebook: CPU: Intel(R) Core (TM) i5-10400 CPU s 2.90GHz 2.90 GHz, GPU: RTX2070 RAM: 16.0 GB, and system type: 64-bit operating system, ×64-based processor (Window 10).

4.1. Experiment 1: Evaluate on HAR. The activity set dataset we use is the 2012 "A public domain dataset for human activity recognition using smartphones" [22].

The data was collected from 30 subjects between the ages of 19 and 48, who wore a waist smartphone that recorded motion data and performed one of six standard activities. Videos of each theme performing activities are recorded, and motion data is manually marked from those videos. The six activities performed are as follows: (1) walk, (2) go upstairs, (3) go downstairs, (4) sit, (5) stations, and (6) lie down. Smartphones record accelerometer and gyroscope data for x, y, and z, with observations recorded at 50 Hz [6].

The original dataset was not available, and we used preprocessed data. Pretreatment includes preprocessing accelerometers and gyroscopes using noise filters. Split the data into 2.56 seconds (128 data points) each, overlapping by 50%. Split the accelerometer data into gravity (total) and body movement components [5].

The dataset can be loaded free of charge from the UCI machine learning repository. The dataset website is as follows: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones.

Each experimental data is with 128 time points and 9 channels. The number of training sample is 2947, and the number of the whole sample is 7,352. No validation sets are divided. Figure 3 is the architecture used by the MCL model in activity HAR. And Table 2 shows the superparameter settings of MCL based on HAR.

The experiment results based on HAR are shown in Table 3. We can see that the MCL model has a higher accuracy and better performance on activity aware (HAR) than the general ConvLSTM model. MCL does not have much computational power in terms of computational costs.

4.2. Experiment 2: Evaluate on MIT-BIH. We chose MIT-BIH Arrhythmia Database data, which is the most commonly used database in the ECG space. The database is available for free with PhysioNet. The website is as follows: https://www.physionet.org/cgi-bin/atm/ATM.

The MIT-BIH Arrhythmia Database contains 48 double-lead ECG records, of which, with the exception of a few records, the first lead for each record is an II lead, each recording 30 minutes long with a sample rate of 360 Hz. Each record has 650,000 messages. We make small wave denoising of the data and perform heart beat interception on the basis of R peak, using each beat as a piece of data. At the same time, we divide all heart beats into four categories: "Normal (N)," "Left Beam Block (LBBB)," "Right Beam Block (RBBB)," and "Room Early Fight (PVC)" for training [20].

A training sample is a heartbeat with 256 time points, a single channel, and each sample belongs to one of four types of heart beats. The number of training sample is 80538. The number of the whole sample is 20134. And Table 4 shows the superparameter settings of MCL based on MIT-BIH.

The experiment results based on MIT-BIH are shown in Table 5. MCL and Multi_CNN perform little differently in data classification training for MIT-BIH Arrhythmia Database. For data sets large enough, MCLs cannot go any further when the average CNN model can reach a theoretical limit close to human performance.

4.3. Experiment 3:Evaluate on Sleep-EDF. The Sleep-EDF database contains 197 multisonic graphical sleep records, including EEG, EOG, EMG, and event tags. Some records also include breathing and body temperature. We make sleep state classification predictions. The input ECE signals are classified into one of five categories: wake (W), nonfast eye movement (N1, N2, N3), and fast eye movement (REM). The Sleep-EDF Public Database can be downloaded from the website [21]: https://www.physionet.org/content/Sleep-EDFx/1.0.0/.

EOG and EEG signals are sampled at 100 Hz, respectively. The EMG signal is filtered through electron high pass filtering, correction, and low pass and then samples the EMG envelope in root squares at 1 Hz. We divided the data into 42,308, each with 3,000 time series, a single channel, and each sample representing a brainwave signal.

Table 2: Superparameter settings based on HAR.

ConvLSTM2D	Head one	Head two	Head three
Filters	128	128	128
Kernel_size	(1, 3)	(1, 5)	(1, 7)
Dropout	0.5	0.5	0.5
L2	0.005	0.005	0.005

Table 3: Experiment results based on HAR.

	Accuracy	F1_socre	Recall	Precision
MCL	93.553%	93.503%	93.571%	93.823%
Multi_CNN	91.076%	91.153%	91.459%	91.651%
ConvLSTM	91.805%	91.766%	91.873%	91.893%

Table 4: Superparameter settings based on MIT-BIH.

ConvLSTM2D	Head one	Head two	Head three
Filters	32	32	32
Kernel_size	(1, 3)	(1, 5)	(1, 7)
Dropout	0.5	0.5	0.5
L2	0.005	0.005	0.005

Table 5: Experiment results based on MIT-BIH.

	Accuracy	F1_socre	Recall	Precision
MCL	99.257%	96.519%	95.232%	97.913%
Multi_CNN	99.237%	96.589%	95.807%	97.462%
ConvLSTM	99.036%	95.731%	94.834%	96.771%

Table 6: Superparameter settings based on Sleep-EDF.

ConvLSTM1D	Head one	Head two	Head three
Filters	128	128	128
Kernel_size	7	11	13
Dropout	0.5	0.5	0.5
L2	0.001	0.001	0.001

Table 7: Experiment results based on Sleep-EDF.

	Accuracy	F1_socre	Recall	Precision
MCL	83.016%	75.555%	75.500%	75.742%
Multi_CNN	76.752%	68.302%	68.631%	68.917%
ConvLSTM	82.165%	74.436%	74.819%	74.634%

The sample number of *W*, N1, N2, N3, and REM is 8285, 2804, 17799, 5703, and 7717, respectively. We treat the data unbalanced and adjust the sample weights (2, 6, 1, 3, 2) to reduce overfitting. After preprocessing the data, put it into the MCL model for training. The results of the experiment are as follows:

The total data sample is 42308. We divide the data into training sets (60%), validation sets (20%), and test sets (20%).And Table 6 shows the superparameter settings of MCL based on Sleep-EDF.

The experiment results based on Sleep-EDF are shown in Table 7. MCL performs better in situations where data is difficult to classify.

4.4. Discussion. In most cases, the MCL model has better performance, both in accuracy and on F1_socre, recall, and precision with greater accuracy. Especially for sample data with a small number of timing characteristics, such as HAR in experiment 1, this is because the Multi_head structure in the MCL can extract more features and make more accurate judgments. The MCL model does not have a particular advantage over data sets where general model training results are close to human performance.

5. Conclusion

In this article, we present the multiple-head convolutional LSTM (MCL), a healthcare timing analysis model based on AI technology. We use regularization, data normalization, trimming, and changing sample weights to reduce overfitting and process data imbalances. In the experiment, we also tried to use L1 regularization, increase filters, add more timing inputs, and add more ConvLSTM layers to One_head ConvLSTM, but found that these increased training time and calculation costs, but did not significantly increase mcCs *L* performance; so, we put only one ConvLSTM layer in One_head ConvLSTM to represent MCL performance. In future experiments, we hope to build more complex and in-depth models to better classify the health care timing.

Data Availability

The labeled datasets used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

No potential conflict of interest was reported by the authors.

References

[1] M. Irfan and I. A. Hameed, "Deep learning based classification for healthcare data analysis system," in *2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC)*, pp. 1–6, Krakow, Poland, 2017.

[2] D. H. KISA, M. A. OZDEMIR, O. GUREN, and A. AKAN, "EMG based hand gesture classification using empirical mode decomposition time-series and deep learning," in *2020 Medical Technologies Congress (TIPTEKNO)*, pp. 1–4, Antalya, Turkey, 2020.

[3] M. Hartmann, H. Farooq, and A. Imran, "Distilled deep learning based classification of abnormal heartbeat using ECG data through a low cost edge device," in *2019 IEEE symposium on computers and communications (ISCC)*, pp. 1068–1071, Barcelona, Spain, 2019.

[4] T. F. Romdhane and M. A. Pr, "Electrocardiogram heartbeat classification based on a deep convolutional neural network

and focal loss," *Computers in Biology and Medicine*, vol. 123, p. 103866, 2020.

[5] https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification.

[6] https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/.

[7] https://www.tensorflow.org/tutorials/structured_data/imbalanced_data.

[8] X. Shi, Z. Chen, and H. Wang, "Convolutional LSTM network: a machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[9] A. Guo, S. Smith, Y. M. Khan, J. R. Langabeer II, and R. E. Foraker, "Application of a time-series deep learning model to predict cardiac dysrhythmias in electronic health records," *PLoS One*, vol. 16, no. 9, article e0239007, 2021.

[10] G. Q. Ali and H. Al-Libawy, "Time-series deep-learning classifier for human activity recognition based on smartphone built-in sensors," *Journal of Physics: Conference Series*, vol. 1973, no. 1, article 012127, 2021.

[11] N. Sridhar, A. Shoeb, P. Stephens et al., "Deep learning for automated sleep staging using instantaneous heart rate," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–10, 2020.

[12] J. Goschenhofer, F. M. J. Pfister, and K. A. Yuksel, "Wearable-based Parkinson's disease severity monitoring using deep learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 400–415, Springer, Cham, 2020.

[13] D. S. Kermany, M. Goldbaum, W. Cai et al., "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131.e9, 2018.

[14] L. M. Hadjiiski and G. D. Tourassi, "Deep learning with non-medical training used for chest pathology identification," in *Medical Imaging 2015: Computer-Aided Diagnosis*, 2015.

[15] D. Shen, G. Wu, and H. I. Suk, "Deep learning in medical image analysis," *Annual Review of Biomedical Engineering*, vol. 19, no. 1, pp. 221–248, 2017.

[16] A. Alharbi, W. Alosaimi, R. Sahal, and H. Saleh, "Real-time system prediction for heart rate using deep learning and stream processing platforms," *Complexity*, vol. 2021, 9 pages, 2021.

[17] D. Suryani, M. N. Shoumi, and R. Wakhidah, "Object detection on dental x-ray images using deep learning method IOP conference series: materials science and engineering," *IOP Publishing*, vol. 1073, no. 1, article 012058, 2021.

[18] D. Bian, P. Mehta, and N. Selvaraj, "Respiratory rate estimation using PPG: a deep learning approach," in *2020 42nd annual international conference of the IEEE engineering in Medicine & Biology Society (EMBC)*, pp. 5948–5952, IEEE, 2020.

[19] https://www.jianshu.com/p/54213477fba7.

[20] https://blog.csdn.net/u011538734/article/details/73477421.

[21] https://www.physionet.org/content/Sleep-EDFx/1.0.0/.

[22] D. Anguita, A. Ghio, and L. Oneto, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, pp. 437–442, 2013.