



Research article

Audio signal-driven non-invasive pneumonia diagnosis: an LSTM model with chaos optimization

Zhou Yang and Lijun Pei*

School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China

* **Correspondence:** Email: peilijun@zzu.edu.cn.

Abstract: Pneumonia is a disease with a high mortality rate worldwide. Existing diagnostic methods for pneumonia inevitably cause varying degrees of trauma to the human body and are time-consuming. To address these drawbacks, this study attempts to use sound signals for pneumonia diagnoses, thereby innovatively integrating acoustic features with machine learning algorithms to establish a non-invasive pneumonia screening method. The experiment utilizes a publicly available respiratory sound dataset containing acoustic samples from pneumonia patients and healthy individuals. By combining Mel-frequency cepstral coefficients (MFCC) with time-domain energy features, a multidimensional feature set is constructed, and five machine learning models, including the backpropagation neural network (BP), the recurrent neural network (RNN), and the long short-term memory network (LSTM), are employed for binary classification modeling, training, and evaluation. To ensure a sufficient generalization ability, a strategy of ten repetitions of ten-fold cross-validation is introduced, and both chaotic optimization and Bayesian optimization are used to search for the optimal hyperparameters of the models. The results show that among the five baseline models, the LSTM performs the best on the dataset, thereby achieving an accuracy of 91.12% on the validation set. After chaotic optimization, the classification performance of the LSTM is significantly improved, with various metrics enhanced to different extents: for example, the accuracy reaches 97.49% and the F1-score increases to 98.17%, significantly outperforming the results achieved through Bayesian optimization. This method successfully overcomes the drawbacks of traditional pneumonia diagnosis approaches, enables non-invasive screening for asymptomatic or early-stage pneumonia, and offers the diagnostic advantages of high efficiency and low cost, thus providing a new solution for the clinical diagnosis of pneumonia.

Keywords: non-invasive pneumonia diagnosis; sound signal processing; MFCC features; machine learning; hyperparameter optimization

1. Introduction

Pathogens in the air that cause pneumonia enter the lungs through the respiratory tract and infect the lung tissue. Once the pathogens reach the lungs, the local immune system in the human body immediately responds. Immune cells in the alveoli, such as macrophages and dendritic cells, recognize and engulf the pathogens and release cytokines (e.g., tumor necrosis factor, interleukins), activating the immune system to eliminate the pathogens. At the same time, this clearance process is often accompanied by an intensification of the local inflammatory response, which allows inflammatory mediators to enter the bloodstream and may extend the inflammatory response from a local to a systemic level. A patient's symptoms include fever, chills, tachycardia, and so on.

According to reports from the World Health Organization (WHO), approximately 3 million people worldwide die from pneumonia each year, and particularly in developing countries, pneumonia is one of the leading causes of death in children. In 2019, pneumonia caused over 3 million deaths globally and was hailed as the “world's biggest killer”. However, most current diagnostic methods for pneumonia are invasive and can cause varying degrees of trauma and harm to the human body.

To date, the diagnosis of pneumonia mainly involves taking a medical history and imaging examinations, the latter including chest computed tomography (CT), chest X-rays, blood tests, and sputum cultures. However, it is undeniable that these methods all have certain limitations and drawbacks. For example, both X-ray and CT examinations expose patients to radiation, which may pose potential health hazards; sputum cultures and blood tests have relatively long turnaround times, which may lead to delayed diagnoses and consequently affect timely treatments for the patients.

The widespread application of machine learning technologies across various fields has unlocked the potential of many industries. In the medical field, studies utilizing machine learning techniques for the diagnosis of major diseases have emerged in abundance, demonstrating the outstanding classification and diagnostic capabilities of machine learning for diseases. Therefore, the in-depth development and application of machine learning technologies can help advance diagnostic methods for major diseases.

In the past, diagnosing pneumonia required blood draws for hematological tests or imaging examinations such as chest CT, all of which can cause varying degrees of trauma to the patient. For this reason, we have turned to an entirely new, non-invasive diagnostic tool that is low-cost, easily accessible, and does not affect the subject in any way during sampling: sound. It is well known that the lungs are the organs responsible for gas exchange between the body and the external environment. Pulmonary lesions in pneumonia patients can affect their breathing and, in turn, their voice production. Therefore, we hypothesized that sound can reflect lung pathology, and on this basis, we established a novel non-invasive detection method for pneumonia. Specifically, by leveraging machine learning techniques, the probability of a person having pneumonia can be efficiently detected using only their sound samples, thus providing a reference for subsequent clinical judgment. The main contributions of this study are as follows:

- 1) Non-invasive diagnosis: diagnosing pneumonia patients using sound signals avoids traditional procedures such as blood tests, offering a solution for non-invasive pneumonia screening.
- 2) Low diagnostic cost: compared with other test samples, a patient's sound samples are easy to obtain and incur very low costs.
- 3) Convenience and ease of promotion: the non-invasive diagnostic method proposed in this study is remarkably straightforward; simply by inputting a person's sound samples, the presence of pneumonia

can be efficiently detected. This makes it readily deployable in areas with limited medical resources.

4) Reference value: after experimental training and model evaluation, the chaotic-optimization-enhanced long short-term memory network (LSTM) model in this study demonstrates a considerable diagnostic capability for pneumonia, thereby achieving an accuracy of 97.49%. Therefore, the method holds certain reference value and can assist clinicians in the diagnosis, providing a supplementary reference for their judgment.

This paper is divided into six sections: Section 2 reviews other studies on the use of machine learning for diagnosing various diseases; Section 3 describes the acquisition and subsequent processing of the model dataset as well as the methods used for model training; Section 4 introduces the models and the approaches for evaluating model performance; Section 5 presents the results analysis and discussion; and Section 6 summarizes this study.

2. Literature review

To date, numerous scholars have achieved remarkable results in using machine learning methods for the prediction and diagnosis of various diseases.

Asif et al. employed an ensemble voting classifier (both hard and soft voting versions) to predict cardiovascular disease, thereby achieving a prediction accuracy of 92% [1]. In addition, the AdaBoost algorithm performed excellently, thereby achieving a precision of 0.938 and a specificity of 0.926. Gomathy and Naidu combined the Naïve Bayes algorithm with the K-nearest neighbors (KNN), logistic regression, and decision tree algorithms to predict multiple diseases including diabetes, malaria, jaundice, dengue fever, and tuberculosis, thereby successfully attaining an accuracy of 98.95% [2]. Parthiban and Srivatsa utilized the support vector machine (SVM) and Naïve Bayes algorithms to accurately diagnose the risk of heart disease in diabetic patients, with the SVM model achieving an accuracy of 94.60% [3]. Patel and Tejal Upadhyay led a team using decision tree algorithms (including J48, logistic model tree, and random forest) for their research, and the experimental results indicated that the J48 algorithm performed best in heart disease prediction, with an accuracy of 56.76% [4]. Shahbaz et al. employed a generalized linear model for the staging and classification diagnosis of Alzheimer's disease, thereby obtaining an accuracy of 88.24% [5]. Nishat et al. diagnosed chronic kidney disease using the random forest algorithm, thereby achieving a prediction accuracy of up to 99.75% [6]. Shukur and Mijwil combined methods such as SVM, random forest, artificial neural network, logistic regression, and KNN to analyze data from the Cleveland Clinic dataset, ultimately finding that the SVM method performed the best with an accuracy of 96% [7]. Pingale et al. employed machine learning methods including Naïve Bayes classifier, KNN, and decision trees to successfully predict multiple diseases such as diabetes, malaria, jaundice, dengue fever, and tuberculosis based on the patient's symptoms, thereby achieving an accuracy of 94.8% [8].

3. Dataset and sound signal processing

The research methodology of this experiment includes dataset description, data preprocessing, data splitting, oversampling, model training, and evaluation. Figure 1 illustrates the entire process from acquiring sound signals to evaluating the model performance.

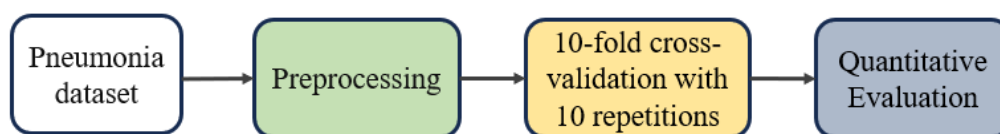


Figure 1. Experimental flowchart.

3.1. Dataset introduction

The audio dataset used in this study is a publicly available dataset from Kaggle (Asthma Detection Dataset Version 2) [9]. This dataset was created by processing and segmenting recordings from two standard audio datasets, together with additional recordings from individual patients. The entire dataset contains a total of 1211 audio samples, divided into five distinct categories representing different lung conditions: 288 asthma samples, 104 bronchitis samples, 401 chronic obstructive pulmonary disease (COPD) samples, 255 pneumonia samples, and 133 healthy samples. For this study, a total of 418 samples were used, comprising the 285 pneumonia patient audio samples and 133 healthy audio samples from the dataset. The sources of the dataset are as follows:

1) Respiratory sound database: approximately 170 samples were obtained from this database, preprocessed to remove noise, and segmented into sections of 5–6 seconds each. The original database can be accessed at: [10.1016/j.bbe.2020.11.003](https://doi.org/10.1016/j.bbe.2020.11.003).

2) International conference on biomedical and health informatics (ICBHI) dataset: this dataset contributed 212 samples and is well known for its comprehensive collection of respiratory sounds. These were also preprocessed and segmented. The dataset is available from the PhysioNet ICBHI repository.

3) Additional samples from 18 patients: the remaining samples were collected from 18 individual patients, including normal (healthy) individuals and patients diagnosed with various respiratory diseases such as asthma, bronchitis, COPD, and pneumonia.

In addition, the dataset underwent preprocessing steps including data cleaning, amplification, and segmentation. Data cleaning refers to the use of the WavePad software for noise reduction and cleaning to enhance the clarity of lung sounds. Irrelevant, incomplete, or noisy entries were removed. Amplification refers to adjusting the recording volume using WavePad's amplification function to ensure consistent loudness across all samples. Segmentation refers to splitting the audio files to ensure uniform duration, thereby facilitating consistency in analysis. All audio files were standardized to 5–6 seconds.

3.2. Sound signal processing

Figure 2 illustrates the relevant operations of audio preprocessing. According to the accompanying description of the dataset, all audio files were subjected beforehand to data cleaning, amplification, segmentation, and other processing, and were all standardized to 5–6 seconds. Next, we performed

audio feature extraction on the audio files.

Before feature extraction, we first normalized the audio to eliminate amplitude differences in the audio signal, followed by pre-emphasis filtering to enhance the high-frequency components of the audio signal and compensate for the potential high-frequency loss in subsequent processing. The normalization and pre-emphasis filtering formulas are as follows:

$$y_{normalized} = \frac{y}{\max(|y|)}, \quad (3.1)$$

$$y(t) = x(t) - \alpha x(t-1). \quad (3.2)$$

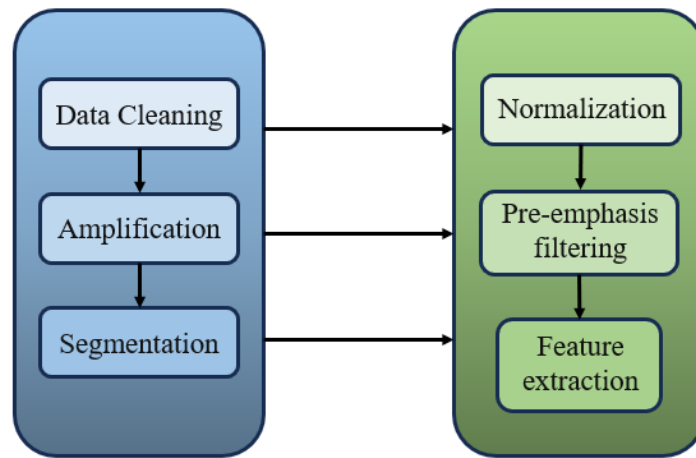


Figure 2. Preprocessing flowchart.

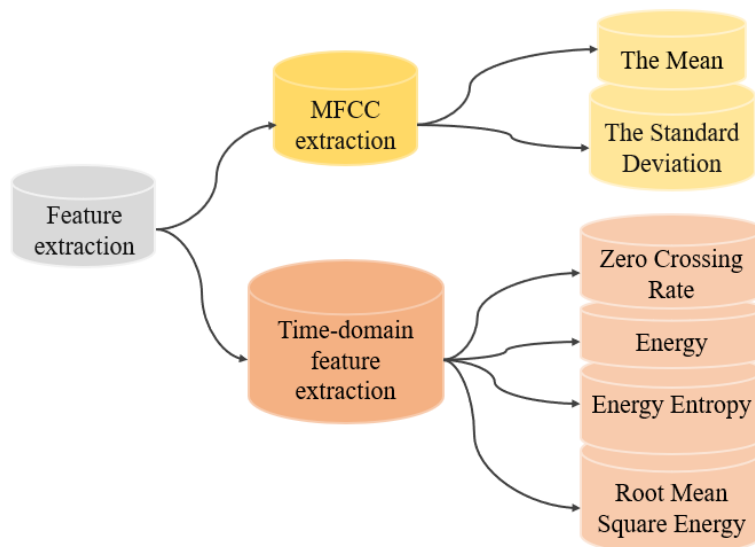


Figure 3. Flowchart of feature extraction.

As shown in Figure 3, when performing feature extraction on the audio files, this study chose to use Mel-frequency cepstral coefficients (MFCC) and several other audio time-domain features, the latter including zero-crossing rate (ZCR), energy, energy entropy, and root mean square (RMS) energy [10].

The MFCC features are obtained by simulating the human ear's perception of frequency to extract the spectral characteristics of the audio signal, and then extracting the audio features of the signal through a cepstral analysis. In this process, the entire audio is divided into several frames. For each frame, multiple MFCC coefficients can be extracted, typically 12 or 13. In this experiment, 13 MFCC coefficients were extracted per frame, and the mean and standard deviation of the k -th MFCC coefficient across all frames ($k = 1, 2, \dots, 13$) were calculated, thus yielding the mean and standard deviation of each of the 13 MFCC coefficients. Through this step, we extracted the average and standard deviation of each coefficient over the entire audio file. Among the time-domain features, the ZCR represents the number of times the signal waveform crosses zero. The ZCR calculation formula is as follows:

$$ZCR = \frac{1}{N-1} \sum_{n=1}^{N-1} I(\text{sign}(x[n]) \neq \text{sign}(x[n-1])), \quad (3.3)$$

where I is an indicator function, which evaluates to 1 if its argument is true, and 0 otherwise, $x[n]$ denotes the n -th sample value of the signal, and N is the number of signal samples. Energy measures the strength of the signal. Portions with a high energy typically represent louder sounds or speech signals. The energy calculation formula is as follows:

$$\text{Energy} = \frac{1}{N} \sum_{n=1}^N x[n]^2. \quad (3.4)$$

Energy entropy measures the complexity of the energy distribution of the signal. The energy entropy calculation formula is as follows:

$$H = - \sum_{i=1}^M \left(\frac{E_i}{\sum_{j=1}^M E_j} \right) \log_2 \left(\frac{E_i}{\sum_{j=1}^M E_j} \right), \quad (3.5)$$

where E_i represents the energy of the i -th frame, M denotes the total number of frames, and $\sum_{j=1}^M E_j$ is the sum of the energies of all frames. RMS Energy is defined as the square root of the mean of the squared signal magnitudes, and it reflects both the strength and variation of the signal. The specific calculation formula is as follows:

$$RMS = \sqrt{\frac{1}{N} \sum_{n=1}^N x[n]^2}. \quad (3.6)$$

After feature extraction, the above 26 MFCC features and 4 time-domain features were combined to obtain a 30-dimensional feature vector. Then, the feature vectors of all audio samples were merged to form a 418×30 feature matrix. Next, Z-score normalization was applied column-wise to this feature matrix (i.e., adjusting the mean of each column to 0 and the standard deviation to 1), ensuring that the distributions of all features are consistent in scale. Let the mean of the i -th column be μ_i and the standard deviation be σ_i . For the feature value x_{ij} in the i -th column and j -th row ($i = 1, 2, \dots, 30$; $j = 1, 2, \dots, 418$), the standardized value is denoted as z_{ij} ($i = 1, 2, \dots, 30$; $j = 1, 2, \dots, 418$), and the calculation formula is as follows:

$$z_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}. \quad (3.7)$$

4. Models and evaluation metrics

The machine learning baseline models used in this study are the backpropagation neural network (BP), recurrent neural network (RNN), LSTM, gated recurrent unit (GRU), and echo state network (ESN). The hyperparameter optimization methods employed include chaotic optimization and Bayesian optimization. The model evaluation metrics used include accuracy, precision, recall, F1-score, as well as the confusion matrix, ROC curve, and F1-score comparison chart.

4.1. Brief introduction to ten repetitions of ten-fold cross-validation

Ten repetitions of ten-fold cross-validation, consisting of ten independent repetitions, is an extended form of standard ten-fold cross-validation.

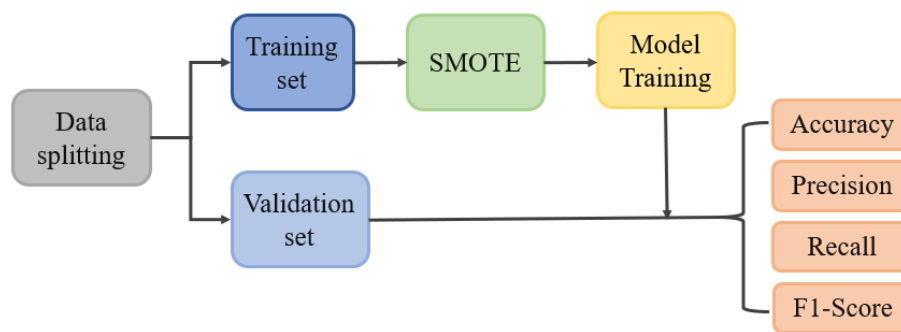


Figure 4. Schematic of one fold in 10×10 -fold cross-validation.

The dataset used in this study suffers from a severe class imbalance problem, with 285 pneumonia samples and only 133 healthy samples. As shown in Figure 4, to eliminate the prediction bias caused by the imbalanced class ratio, the synthetic minority over-sampling technique (SMOTE) was applied to the training set. SMOTE was only used for model training and was not involved in the evaluation on the training and validation sets. The SMOTE algorithm is based on the KNN concept, generating synthetic samples among the minority class samples (healthy samples) to enrich the feature distribution of the minority class. During execution, the parameters of SMOTE were set as follows: random number generator seed (random_state) = 42, k-nearest neighbors parameter (k_neighbors) = 5, sampling strategy (sampling_strategy) = 'auto', where the third parameter functions to automatically balance all class counts to the maximum value.

Ten-fold cross-validation refers to randomly dividing the dataset into ten equal subsets, selecting one subset as the validation set and the remaining nine subsets as the training set, conducting model training and validation, and recording the metric values. This process is called a fold. There are ten folds in total, so this process is repeated ten times, each time selecting a different subset as the validation set. Ten repetitions of ten-fold cross-validation are an extension of ten-fold cross-validation, meaning that the ten-fold cross-validation is carried out ten times, and each time the dataset is randomly re-divided into ten equal parts. Figure 4 illustrates the process of one fold within the ten repetitions of ten-fold cross-validation. In this study, the final performance metric values of the model default to the average over all folds.

4.2. Brief introduction to the models

4.2.1. Backpropagation neural network (BP)

BP is a type of feedforward neural network, and its core idea is backpropagation [11–15]. The BP neural network consists of three layers: an input layer, a hidden layer, and an output layer. During the training process, based on the output error, the BP model can adjust the weights of each neuron through backpropagation. For each input, the BP neural network computes the corresponding output via weighted sums, where commonly used activation functions such as Sigmoid and ReLU are applied at each layer (Figure 5).

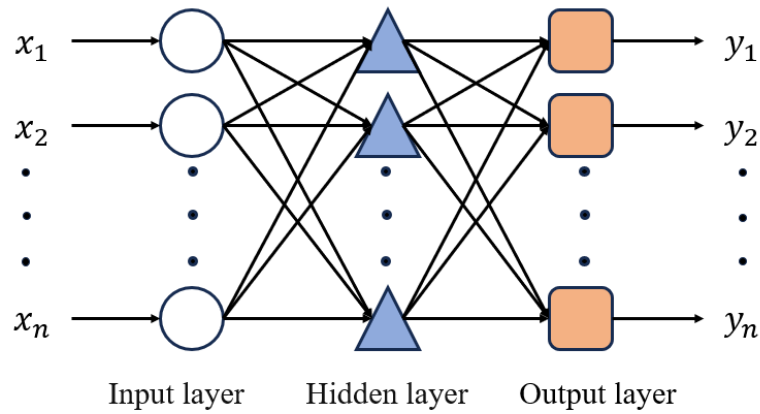


Figure 5. Schematic diagram of the BP architecture [15].

For each input, the BP neural network computes the corresponding output by means of weighted sums, where commonly used activation functions such as Sigmoid and ReLU are applied at each layer, as follows:

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}}, \quad (4.1)$$

$$\text{ReLU}(z) = \max(0, z), \quad (4.2)$$

where z denotes the weighted sum of the input values. Then, the initial output error value L is calculated using a loss function. Assuming that the value of a neuron after the activation function is a_1 and its input is z_1 , the gradient of the loss function with respect to the weight of this neuron can be obtained as follows:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w}. \quad (4.3)$$

The weight parameters of each neuron are updated according to the gradient descent method, as follows:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial L}{\partial w}. \quad (4.4)$$

Through continuous iterative adjustments, the model performance achieves the desired level.

4.2.2. Recurrent neural network (RNN)

RNN is a neural network architecture with memory capability, suitable for processing data with temporal characteristics, such as speech and text [16–20]. RNN also has a three-layer structure consisting of an input layer, a hidden layer, and an output layer. Unlike traditional feedforward neural networks, RNN has a unique internal recurrent connection structure, which makes the current output depend on both the current input and the previous states (Figure 6).

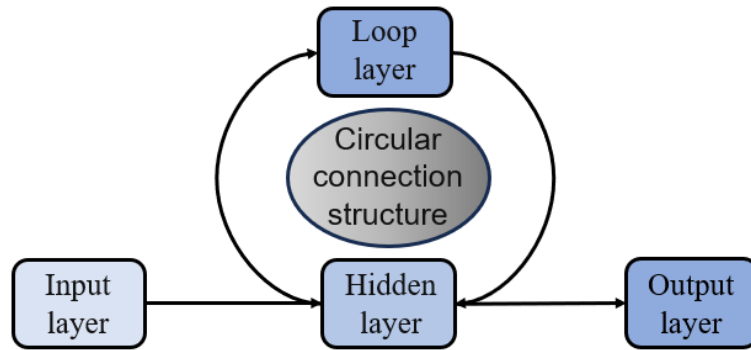


Figure 6. Schematic diagram of the RNN architecture [20].

Let the input at the current time step be x_t , the hidden state (output of the hidden layer) be h_t , and the output be y . The formula to calculate the hidden state is as follows:

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h). \quad (4.5)$$

The final output is obtained through the hidden state, as follows:

$$y = \sigma(w_y h_t + b_y), \quad (4.6)$$

where W_h , U_h , and w_y are weight parameter matrices, b_h and b_y are bias terms, and σ is the activation function.

4.2.3. Long short-term memory (LSTM)

LSTM is an improved RNN model that successfully addresses the vanishing gradient and exploding gradient problems encountered by RNNs when processing long sequences [21–25]. LSTM effectively captures long-term dependencies by introducing three gating mechanisms (input gate, forget gate, and output gate), making it more effective than traditional RNNs in processing temporal data. Based on the previous hidden state h_{t-1} and the current input x_t , the outputs of the input gate, forget gate, and output gate can be obtained through different weight matrices W and bias terms b .

The input gate determines how much information from the current input is used. Given the input gate weight W_i and bias b_i , the output value i_t can be calculated as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (4.7)$$

where the forget gate determines which information to discard. Given the forget gate weight W_f and bias b_f , the output value f_t can be obtained as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad (4.8)$$

where the output gate determines the output of the current hidden state. Given the output gate weight W_o and bias b_o , the final output value o_t can be obtained as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o). \quad (4.9)$$

4.2.4. Gated recurrent unit (GRU)

GRU is also an improved RNN model designed to solve the vanishing gradient problem of RNNs [26–30]. Similar to LSTM, GRU adopts gating mechanisms to determine the storage and updating of information, but its structure is more simplified, with only two gates: the update and the reset gates (Figure 7).

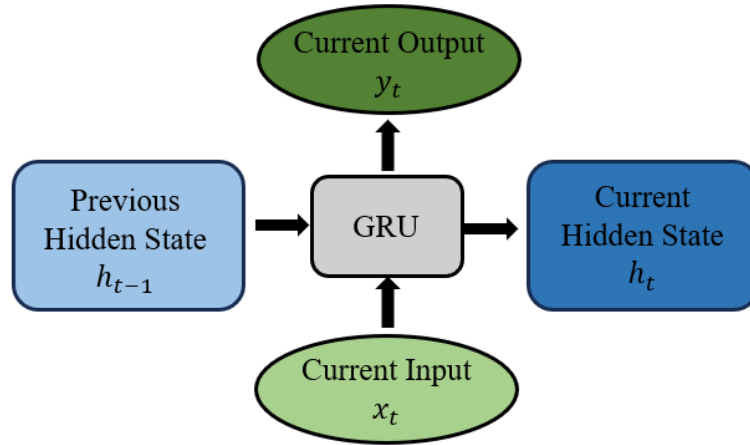


Figure 7. Schematic diagram of the GRU architecture [30].

The update gate determines the update ratio of the hidden state, similar to the combination of the input gate and forget gate in LSTM. Given the weight matrix W_z and bias b_z , and combining the previous hidden state h_{t-1} with the current input x_t , the output z_t of the update gate is obtained as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \quad (4.10)$$

where the reset gate controls the degree to which the current input information is combined with the previous hidden state. Similar to the logic of the update gate, given the weight matrix W_r and bias b_r , and combining the previous hidden state h_{t-1} with the current input x_t , the output r_t of the reset gate is obtained as follows:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r). \quad (4.11)$$

The output value of the update gate $z_t \in [0, 1]$ represents the degree to which the hidden state is updated at the current time step: if z_t is close to 1, then it means that the hidden state is almost

completely updated; and conversely, if it is close to 0, then the hidden state is almost entirely retained. For the reset gate, its output value is $r_t \in [0, 1]$: if r_t is close to 1, then it means that the previous hidden state has almost entirely preserved; and if it is close to 0, then the previous hidden state has almost no influence on the current hidden state.

4.2.5. Echo state network (ESN)

ESN is a special type of neural network model with a recurrent structure [31–35]. Its structure includes an input layer, a reservoir, and an output layer. The reservoir is randomly generated, and the connection weights between neurons within it are randomly initialized and do not change during training. Therefore, in the ESN architecture, only the weights of the output layer need to be trained, while the other structures are fixed and do not require training (Figure 8). As a result, compared with RNNs, the training process of the ESN is relatively simple and stable.

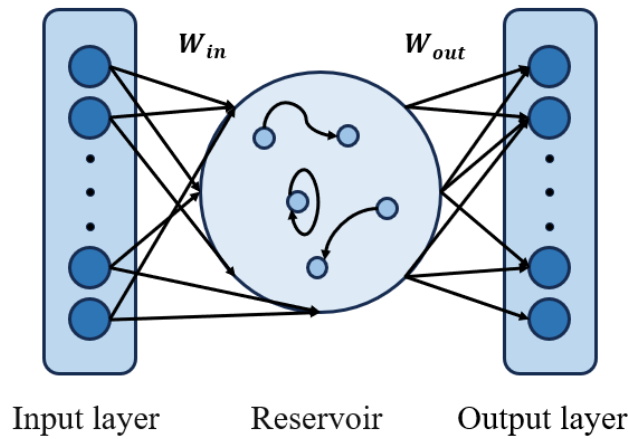


Figure 8. Schematic diagram of the ESN architecture [35].

In the computation of the output layer, the output at time t is y_t , and the calculation formula is as follows:

$$y_t = W_{out}x_t, \quad (4.12)$$

where W_{out} is the connection weight matrix of the output layer, which needs to be obtained through training.

4.3. Hyperparameter optimization

4.3.1. Chaos optimization

Chaotic optimization is a hyperparameter optimization method based on the chaos theory. It utilizes the dynamic behavior of chaotic systems to generate efficient parameter search sequences, thereby finding the globally optimal parameters. This study employs a chaotic hyperparameter optimization algorithm based on the Logistic map as follows:

$$x_{n+1} = r \cdot x_n \cdot (1 - x_n), \quad (4.13)$$

where the initial value $x_0 \in (0, 1)$ and r is the control parameter. Research has found that when $r = 4$, the map enters a chaotic state, capable of generating chaotic sequences. It can search for parameters over a relatively wide region with good ergodicity, effectively avoiding local optima and thus helping the model find the globally optimal parameters.

4.3.2. Bayesian optimization

Bayesian optimization is a global optimization method based on a probabilistic model, primarily used for optimizing expensive, non-differentiable, or irregular black-box functions. A Gaussian process (GP) is typically used as the surrogate model, as follows:

$$f(x) \sim GP(m(x), k(x, x')). \quad (4.14)$$

This surrogate model is used to predict the value of the objective function. We choose the commonly used expected improvement function as the acquisition function, with the formula as follows:

$$EI(x) = E[\max(f(x) - f^+, 0)], \quad (4.15)$$

where f^+ is the current best objective value. Subsequently, the next evaluation point x_{next} is selected, $f(x_{next})$ is computed through the true objective function, and the result $(x_{next}, f(x_{next}))$ is added to the sample set to update the Gaussian process. The model is continuously adjusted using new evaluation points until the stopping condition is met, at which point the optimal hyperparameters required by the model are obtained.

4.3.3. Optimization objective and parameter space

To find the optimal model performance, this study employs chaotic optimization and Bayesian optimization algorithms to perform hyperparameter optimization on the LSTM. The objective function used for optimization is the binary cross-entropy loss (BCE Loss) of the model on the validation set. The hyperparameters to be optimized include the learning rate (lr), hidden layer dimension (hidden_dim), early stopping patience (patience), and batch size (batch_size).

Here, the learning rate (lr) controls the step size of weight updates. The hidden layer dimension (hidden_dim) determines the dimensionality of the internal state of the LSTM units; a dimension that is too small may lead to underfitting, whereas a dimension that is too large may cause overfitting and significantly increase computational cost. The early stopping patience (patience) refers to the number of consecutive epochs allowed to continue training without termination when the validation loss does not improve, and this parameter can effectively prevent overfitting. The batch size (batch_size) refers to the number of training samples that pass through the network before the model parameters are updated.

4.4. Evaluation metrics

Evaluation metrics are quantitative standards used to measure the model performance, and different metrics are adopted for different problems. As this experiment involves a classification problem, the evaluation metrics we use include accuracy, precision, recall, and F1-score. In addition, confusion matrices, receiver operating characteristic (ROC), and F1-score comparison charts are plotted as graphical metrics to further evaluate the model performance.

TP: true positive—the number of samples correctly predicted as the positive class by the model.

TN: true negative—the number of samples correctly predicted as the negative class by the model.

FP: false positive—the number of samples incorrectly predicted as the positive class by the model.

FN: false negative—the number of samples incorrectly predicted as the negative class by the model.

4.4.1. Dataset statistics

Accuracy is one of the most commonly used classification evaluation metrics. It represents the proportion of correctly predicted samples to the total number of samples. The formula for accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (4.16)$$

Precision refers to the proportion of samples that are actually positive among those predicted as positive by the model. It represents the model's accuracy when predicting the positive class. The formula for precision is as follows:

$$Precision = \frac{TP}{TP + FP}. \quad (4.17)$$

Recall is the proportion of positive class samples that the model correctly identifies. It measures the model's ability to recognize the positive class. The formula for recall is as follows:

$$Recall = \frac{TP}{TP + FN}. \quad (4.18)$$

The F1-score is the harmonic mean of precision and recall, comprehensively taking both into account. It is particularly suitable for data with a class imbalance. The formula for the F1-score is as follows:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (4.19)$$

4.4.2. Graphical evaluation metrics: confusion matrix

A confusion matrix is a table used to evaluate the performance of a classification model, thereby visualizing the model's prediction results for each class. In Figure 9, TP in the bottom-right corner represents the number of samples that are actually positive and correctly predicted as positive, FN in the bottom-left corner represents the number of samples that are actually positive but incorrectly predicted as negative, FP in the top-right corner represents the number of samples that are actually negative but incorrectly predicted as positive, and TN in the top-left corner represents the number of samples that are actually negative and correctly predicted as negative.

In this experiment, the positive class is set as “Pneumonia” and the negative class as “Healthy”. The confusion matrices used in this experiment are the cumulative results of the confusion matrices from all folds.

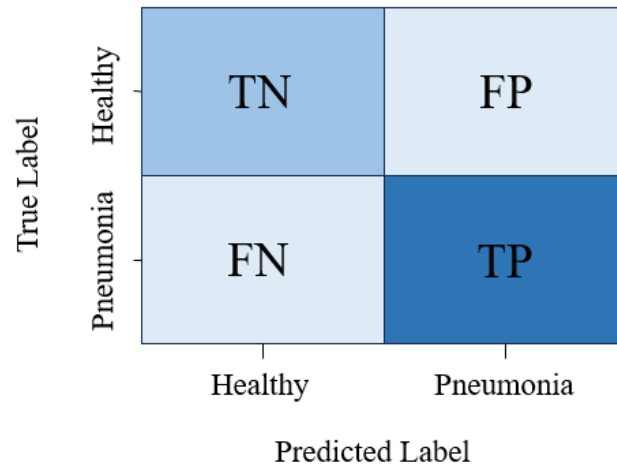


Figure 9. Schematic diagram of a confusion matrix for binary classification [36].

4.4.3. Graphical evaluation metrics: ROC curve

The ROC curve is a commonly used tool to evaluate the classification models. It illustrates the model's performance in a classification task by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The area under the curve (AUC) value is a quantitative measure of the area under the ROC curve. AUC values close to 1 indicate a better classification performance of the model. The horizontal axis of the ROC curve represents the FPR, which indicates the proportion of negative class samples incorrectly predicted as positive by the model. Its calculation formula is as follows:

$$FPR = \frac{FP}{FP + TN}. \quad (4.20)$$

The vertical axis represents the TPR, which has the same definition as recall, indicating the proportion of positive class samples correctly identified by the model. Its calculation formula is as follows:

$$TPR = \frac{TP}{TP + FN}. \quad (4.21)$$

4.4.4. Graphical evaluation metrics: comparison of F1-scores

In this experiment, two curves are plotted in the F1-score comparison chart, representing the training set and validation set data, as shown in Figures 13 and 17. The horizontal axis represents the fold number of the ten repetitions of ten-fold cross-validation, totaling 100 folds, and the vertical axis represents the F1-score of the model on the training set and validation set for each fold. The purpose of plotting this chart is to qualitatively determine whether the model exhibits overfitting. If the training set curve continues to rise while the validation set curve significantly differs from it, or if the F1-score curve of the training set remains high while that of the validation set fluctuates considerably, then it may indicate that the model is overfitting.

Additionally, the experiment provides the metric differences between the model on the training set and the validation set, specifically referring to the differences in the average accuracy, average

precision, etc., as shown in Tables 1 and 2. These data are used to quantitatively analyze whether the model has overfitting. Ultimately, a comprehensive judgment is made based on both qualitative and quantitative analyses to determine whether the model overfits the dataset.

5. Experimental results and analysis

This experiment uses ten repetitions of ten-fold cross-validation to train and validate the models. The classification performance of the five baseline models on the dataset used in this experiment is compared based on the evaluation metrics. The best-performing model among the baseline models is selected for chaotic optimization and Bayesian optimization. By comparing the performances, we can identify the most suitable hyperparameter optimization algorithm for that model.

5.1. Result analysis of the baseline models

5.1.1. Comparison of numerical evaluation metrics

As shown in Table 1 and Figure 10, the model with the best overall performance on the validation set in this experiment is the LSTM. The LSTM achieves an accuracy of 91.12%, a precision of 95.63%, a recall of 91.26%, and an F1-score of 93.25%, all of which are the highest among the five models. For the BP model, although its precision reaches 92.40%, which is relatively good, its accuracy is 81.67%, its recall is 79.91%, and its F1-score is 85.49%, making BP the lowest in terms of overall performance among the five models. Based on the above analysis, the baseline model with the best overall classification performance in this experiment is the LSTM, while the worst is the BP. Therefore, we will not perform visual analysis of graphical metrics for BP in the following sections.

Table 1. Comparison of Results for BP, RNN, LSTM, GRU, and ESN.

Model	Validation Set				Training-Validation Metric Gap			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
LSTM	0.9112	0.9563	0.9126	0.9325	0.0295	0.0207	0.0225	0.0229
RNN	0.8942	0.9424	0.9014	0.9204	0.0231	0.0152	0.0180	0.0177
ESN	0.8535	0.9177	0.8650	0.8890	0.0171	0.0139	0.0096	0.0131
GRU	0.8509	0.9311	0.8495	0.8849	0.0232	0.0159	0.0171	0.0182
BP	0.8167	0.9240	0.7991	0.8549	0.0162	0.0104	0.0128	0.0138

5.1.2. Visualization analysis of graphical metrics: confusion matrix

After ten repetitions of ten-fold cross-validation, each model's confusion matrix is the cumulative result of all folds. In Figure 11, LSTM's class labels are not clearly displayed, so note that label 0 denotes "Healthy" samples and label 1 denotes "Pneumonia" samples. In this confusion matrix, the top-left corner shows correctly predicted "Healthy" samples, the bottom-right shows correctly predicted "Pneumonia" samples, the bottom-left shows "Pneumonia" samples misclassified as "Healthy", and the top-right shows "Healthy" samples misclassified as "Pneumonia".

The confusion matrices in Figure 11 are evaluated and analyzed below. As shown in Figure 11, the correctly identified "Pneumonia" samples are 2601 for the LSTM, 2569 for the RNN, 2465 for the

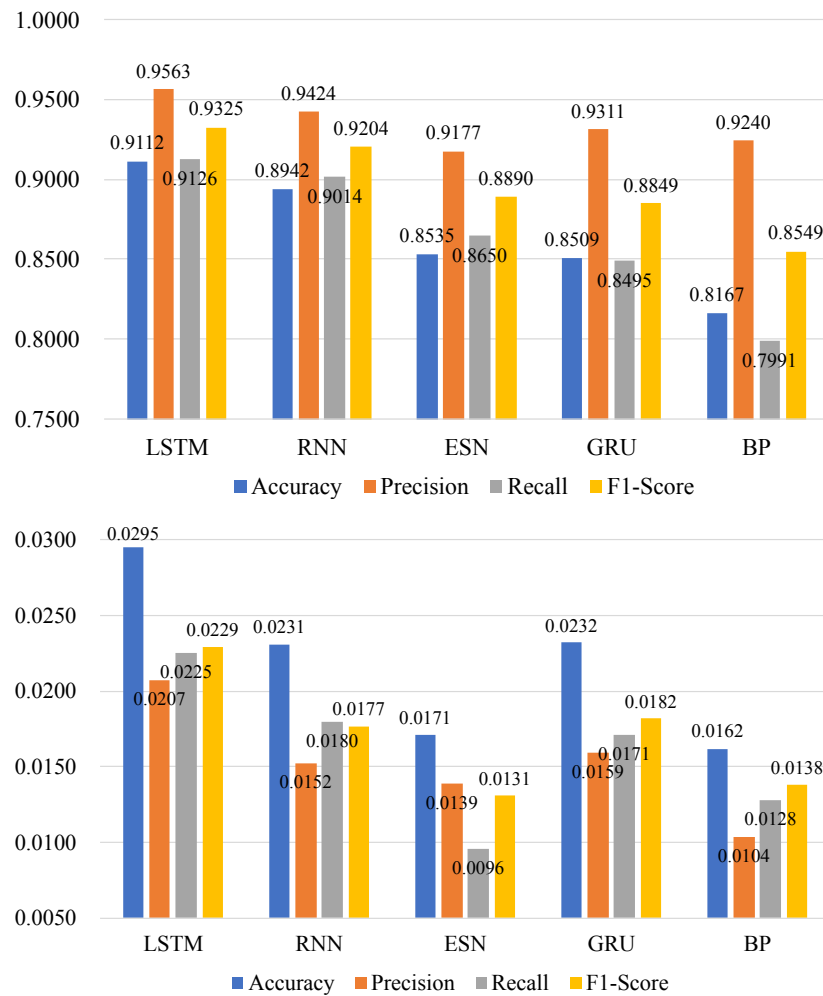


Figure 10. Comparison of results for BP, RNN, LSTM, GRU, and ESN.

ESN, and 2421 for the GRU. This indicates that the LSTM achieves the best performance in recognizing acoustic features of pneumonia patients in this experiment. The numbers of “Pneumonia” samples misidentified as “Healthy” by the four models are as follows: 429 for the GRU, 385 for the ESN, 281 for the RNN, and 249 for the LSTM. Among them, the LSTM has the smallest number of pneumonia patient audio samples incorrectly identified as healthy audio samples, with only 249 cases. The numbers of “Healthy” samples misidentified as “Pneumonia” by the four models are as follows: 227 for the ESN, 194 for the GRU, 161 for the RNN, and 122 for the LSTM. Among them, the LSTM has the smallest number of healthy audio samples incorrectly identified as pneumonia patient audio samples, with only 122 cases. In summary, the LSTM demonstrates the best prediction performance for “Pneumonia” samples in this experiment.

5.1.3. Visualization analysis of graphical metrics: ROC curve

The ROC curves in Figure 12 are the average ROC curves across all folds of the ten repetitions of ten-fold cross-validation. The horizontal axis is the FPR, representing the proportion of negative class

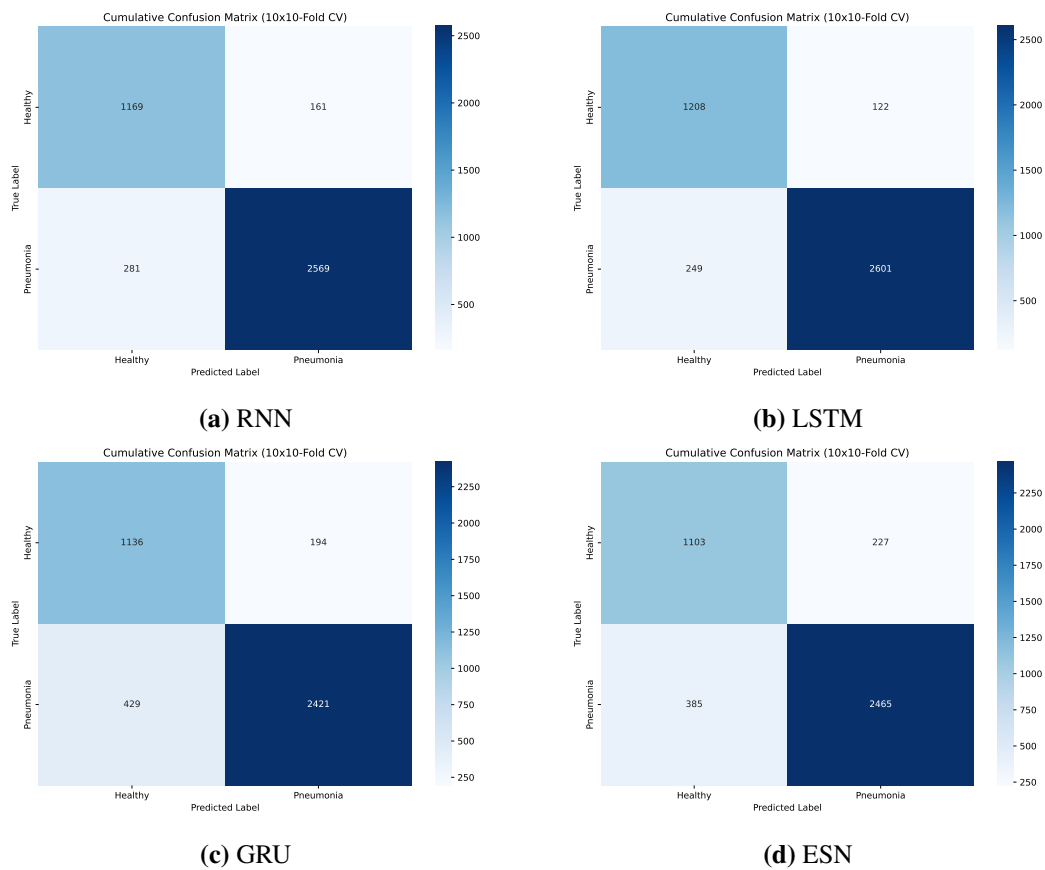


Figure 11. Comparison of confusion matrices for RNN, LSTM, GRU, and ESN.

samples (“Healthy”) incorrectly predicted as the positive class (“Pneumonia”); the vertical axis is the TPR, representing the proportion of positive class samples correctly predicted as positive.

The areas under the ROC curves (AUC values) of the four models, from highest to lowest, are 0.9667 (LSTM), 0.9495 (RNN), 0.9307 (ESN), and 0.9251 (GRU). From the perspective of the ROC curves, the ranking of classification performance from high to low is as follows: LSTM, RNN, ESN, GRU. Therefore, the LSTM exhibits the best classification performance on the pneumonia dataset, followed by the RNN.

5.1.4. Visualization analysis of graphical metrics: F1-score comparison

Table 1 and Figure 13 serve as quantitative and qualitative analyses, respectively, to determine whether the models exhibit overfitting.

When the metric values of the model on the training set are much larger than those on the validation set, it can be preliminarily judged that the model exhibits overfitting. As shown in Table 1, the metric with the largest difference for all four models is accuracy, and the specific differences, from largest to smallest, are 2.95% (LSTM), 2.32% (GRU), 2.31% (RNN), and 1.71% (ESN). From the above data, it can be seen that the maximum metric differences of the four models in this experiment do not exceed 3%, indicating that these four models have good generalization abilities and a low risk of overfitting. Among them, the ESN has the smallest difference of 1.71%, exhibiting the strongest

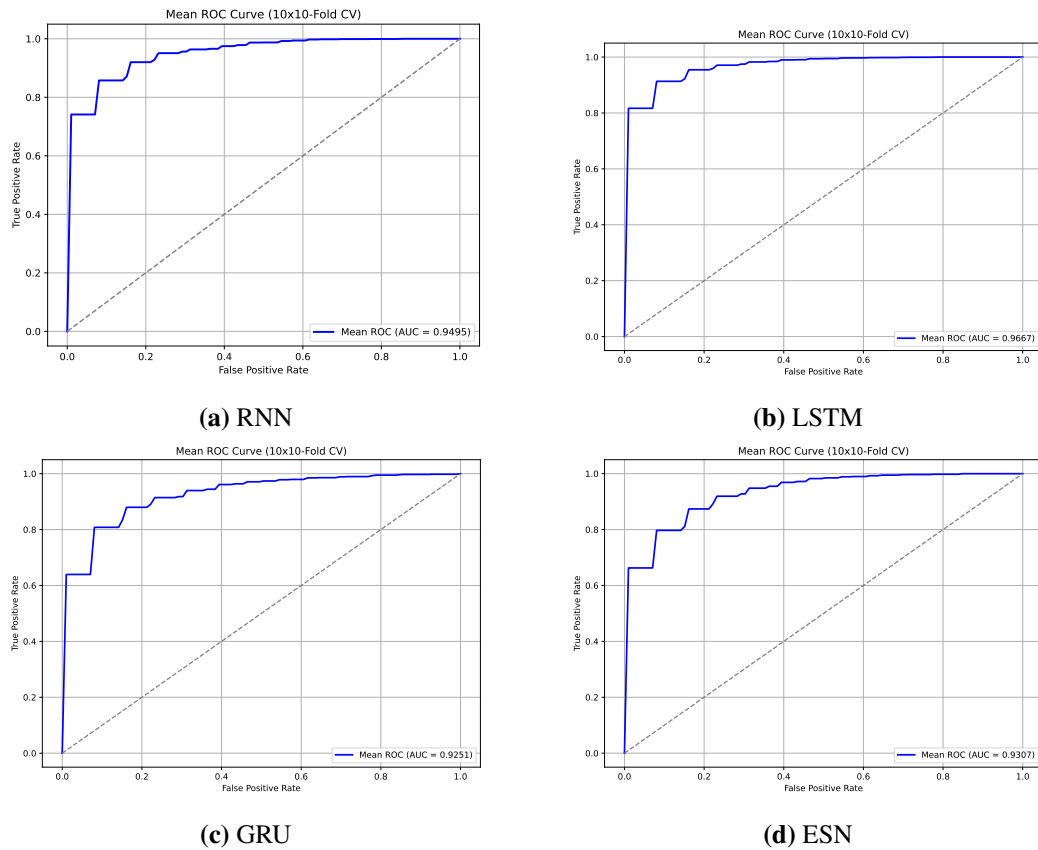


Figure 12. Comparison of ROC curves for RNN, LSTM, GRU, and ESN.

generalization ability among the four models, followed by the RNN and the GRU with differences of 2.31% and 2.32%, respectively, and finally the LSTM with a maximum difference of 2.95%, which, however, does not exceed 3%.

From Figure 13, in the F1-score comparison charts of the RNN, LSTM, GRU, and ESN, the F1-scores for the training set and validation set at each fold are generally close, and the fluctuation trends of the two curves are similar. No cases are found where the F1-score on the training set is significantly higher than that on the validation set, nor is there a progressively widening gap.

Based on the above visual performance analysis and overfitting risk analysis, it can be concluded that in this experiment, the LSTM model achieves a significantly better overall performance than the other three models. Although LSTM's generalization ability is slightly inferior to that of the other three models, its maximum metric difference does not exceed 3%, indicating that the LSTM performs well in terms of overfitting risk. Therefore, we select the LSTM model as the target for the next step of hyperparameter optimization.

5.2. Hyperparameter optimization results of LSTM

The experiment found that among the five baseline models, the LSTM model exhibited the most superior classification performance and good generalization ability. Next, we applied chaotic optimization and Bayesian optimization to the LSTM, and then analyzed and evaluated the

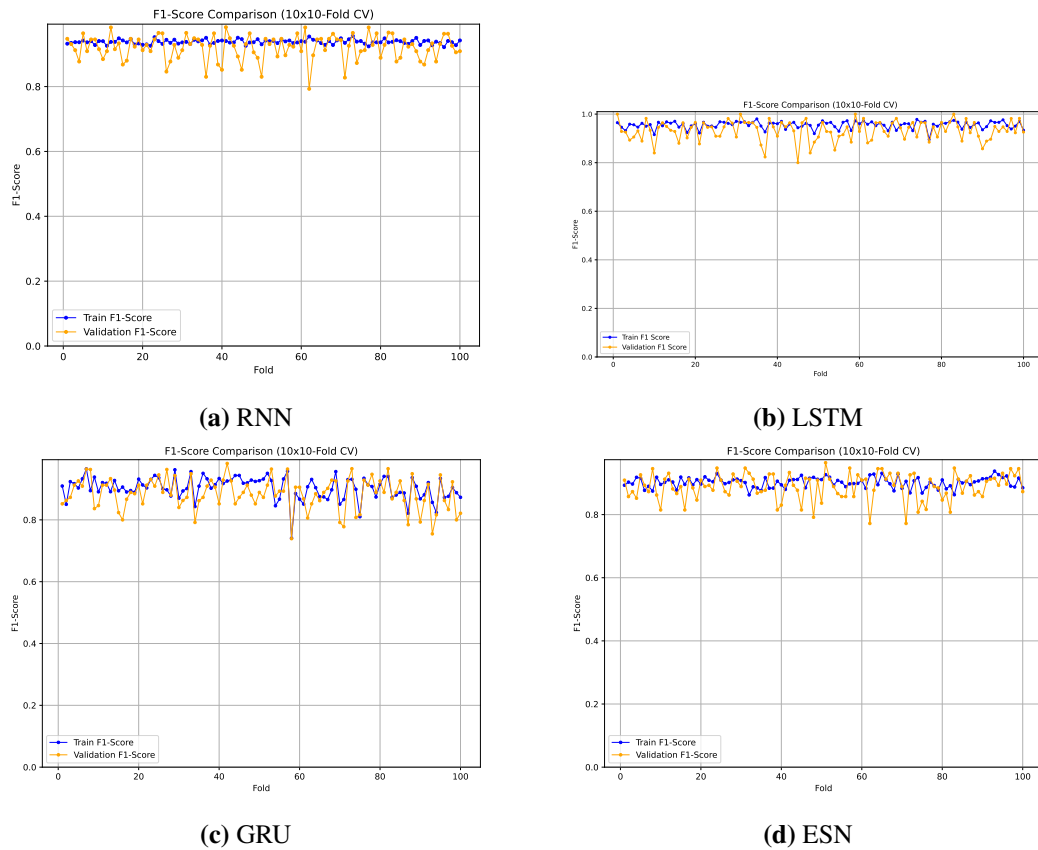


Figure 13. Comparison of F1-scores for RNN, LSTM, GRU, and ESN.

performance of the optimized models. It was ultimately found that, compared with Bayesian optimization, chaotic optimization led to a greater improvement in the overall performance of the LSTM.

The optimal hyperparameters obtained by chaotic optimization were as follows: learning rate (lr) = 1.17×10^{-3} , hidden layer dimension ($hidden_dim$) = 17, early stopping patience ($patience$) = 5, and batch size ($batch_size$) = 21.

5.2.1. Comparison of numerical evaluation metrics

As shown in Table 2 and Figure 14, in this experiment, both chaotic optimization and Bayesian optimization essentially achieved performance improvements for the LSTM. In terms of accuracy, the accuracy of the LSTM after chaotic optimization reached 97.49%, and after Bayesian optimization, it reached 93.04%, representing improvements of 6.37% and 1.92%, respectively, compared with the unoptimized LSTM model, with chaotic optimization achieving the highest accuracy. In terms of precision, the precision of the model after chaotic optimization and Bayesian optimization reached 98.21% and 95.47%, respectively. Compared with the unoptimized LSTM model, chaotic optimization improved this metric by 2.58%, whereas Bayesian optimization caused a decrease of 0.16%, indicating that Bayesian optimization reduced the accuracy of LSTM in predicting the positive class. In terms of recall, the recall of the model after chaotic optimization and Bayesian

optimization was 98.18% and 94.39%, respectively, representing improvements of 6.92% and 3.13% compared with the unoptimized LSTM, with the recall after chaotic optimization being significantly higher than that after Bayesian optimization. In terms of the F1-score, the F1-score of the LSTM after chaotic optimization reached 98.17%, and after Bayesian optimization it reached 94.85%, representing improvements of 4.92% and 1.60%, respectively, compared with the unoptimized LSTM, indicating that the F1-score of LSTM after chaotic optimization is higher than that after Bayesian optimization.

Table 2. Comparison of LSTM Results after Different Optimizations

Model	Validation Set				Training-Validation Metric Gap			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
LSTM	0.9112	0.9563	0.9126	0.9325	0.0295	0.0207	0.0225	0.0229
LSTM+Bayesian	0.9304	0.9547	0.9439	0.9485	0.0258	0.0225	0.0142	0.0190
LSTM+Chaos	0.9749	0.9821	0.9818	0.9817	0.0243	0.0178	0.0172	0.0178

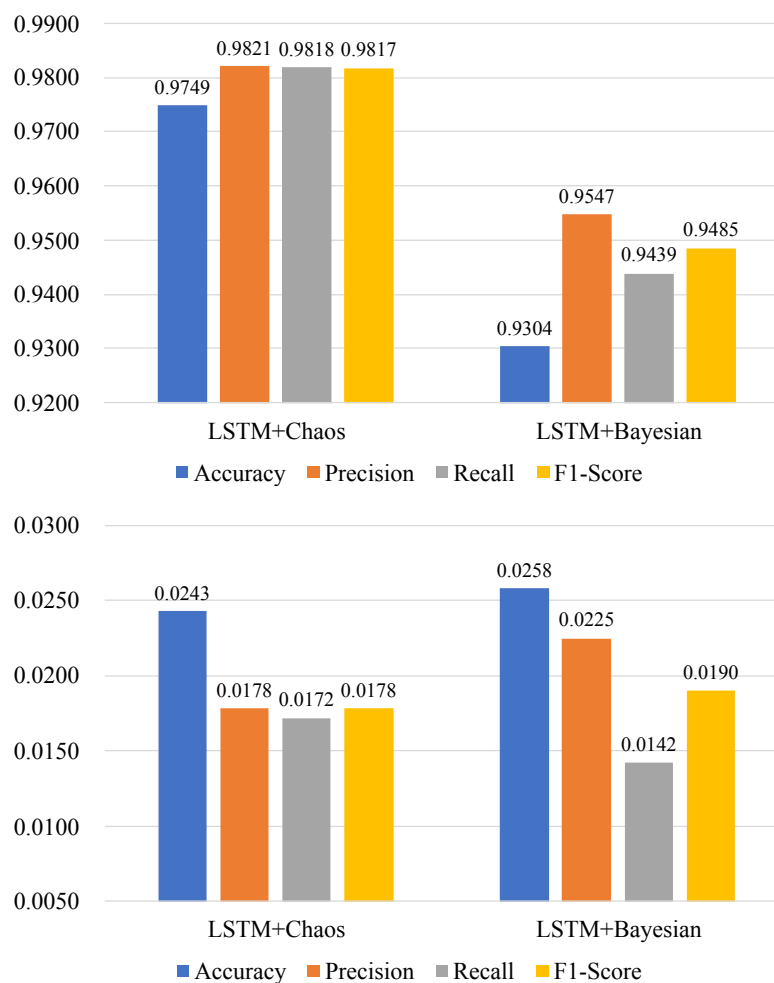


Figure 14. Comparison of LSTM results after different optimizations.

Summarizing the above evaluation, we find that chaotic optimization enables improvements in accuracy, precision, recall, and F1-score of the LSTM model to varying degrees, with the largest improvement in recall, reaching 6.92%. Although Bayesian optimization improves the accuracy, recall, and F1-score of the LSTM model, the improvements are smaller than those from chaotic optimization, and Bayesian optimization causes a decrease in precision instead of an increase. Therefore, in this experiment, the LSTM model after chaotic optimization outperforms the LSTM model after Bayesian optimization and the unoptimized LSTM model across all metrics.

5.2.2. Visualization analysis of graphical metrics: confusion matrix

In Figures 11 and 15, we compare the confusion matrices of the optimized LSTM models with that of the unoptimized LSTM model, and analyze the ability of the LSTM to predict positive class samples. For the LSTM after chaotic optimization, among the samples predicted as positive, 2798 are true positive samples and 53 are negative class samples. Compared with the unoptimized LSTM, the former increases by 197, while the latter decreases by 69. For the LSTM after Bayesian optimization, among the samples predicted as positive, 2690 are true positive samples and 131 are negative class samples. Compared with the unoptimized LSTM, the former increases by 89, while the latter increases by 9. Thus, it can be seen that chaotic optimization improves LSTM's ability to recognize pneumonia audio samples, whereas Bayesian optimization causes a slight decline in LSTM's ability to predict positive class samples. Additionally, this analysis confirms the fact presented in Table 2 that the precision of the LSTM after Bayesian optimization decreased rather than increased.

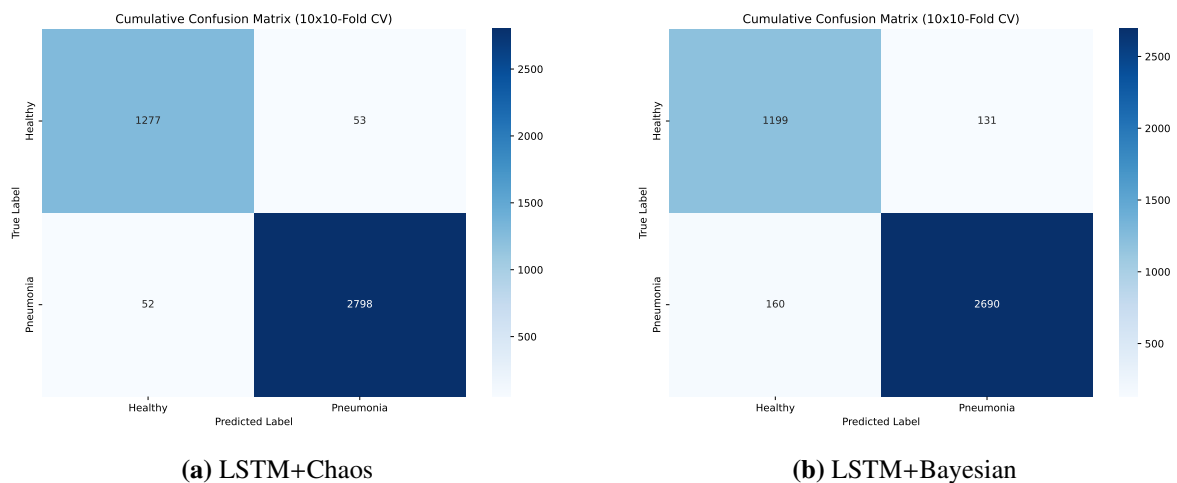


Figure 15. Comparison of confusion matrices for LSTM after different optimizations.

5.2.3. Visualization analysis of graphical metrics: ROC curve

The ROC curves in Figure 16 illustrate the performance of the hyperparameter-optimized LSTM models in the classification task. Combine the two ROC curves in Figure 16 with LSTM's ROC curve in Figure 12 for the performance analysis: for the LSTM model after chaotic optimization, its AUC value is 0.9925; and for the LSTM model after Bayesian optimization, its AUC value is 0.9767. Compared with the unoptimized LSTM model, they represent increases of 0.0258 and 0.01,

respectively, indicating that both optimization methods substantially improved the classification performance of the LSTM model in this experiment, and in this regard, chaotic optimization is slightly superior to Bayesian optimization.

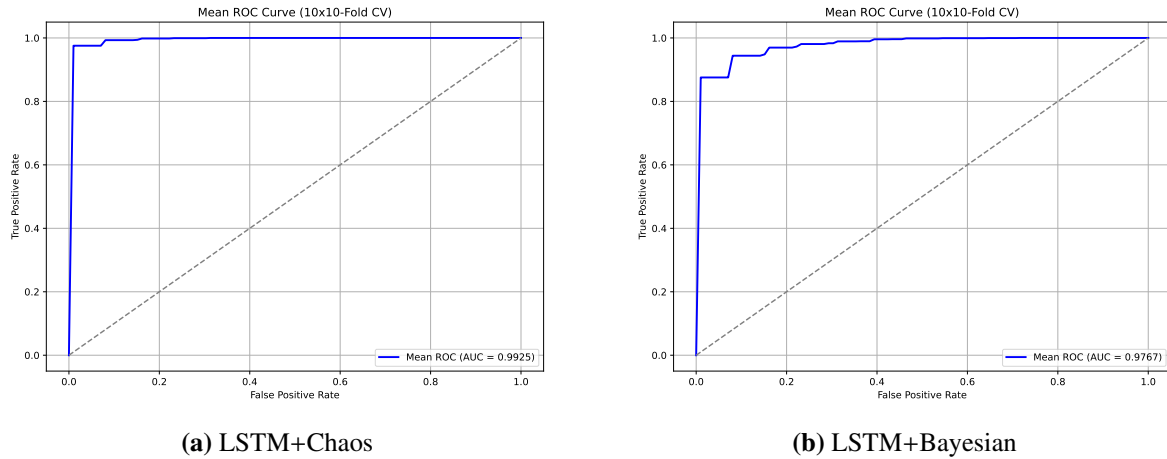


Figure 16. Comparison of ROC curves for LSTM after different optimizations.

5.2.4. Visualization analysis of graphical metrics: F1-score comparison

Table 2 presents the metric differences of the LSTM between the training set and validation set after hyperparameter optimization, and combined with the F1-score comparison chart in Figure 17, an overfitting risk analysis is conducted on the optimized LSTM. The metric with the largest difference for the LSTM model is accuracy, with a difference of 2.95%, which is very close to 3%. After hyperparameter optimization, the metric with the largest difference remains accuracy, and the maximum differences for chaotic optimization and Bayesian optimization are 2.43% and 2.58%, respectively. Compared with 2.95%, these represent reductions of 0.52% and 0.37%, respectively. It indicates that both hyperparameter optimization methods improved the generalization ability of the LSTM and reduced the model's risk of overfitting. On the other hand, as can be seen from Figure 17, after two types of hyperparameter optimization, the F1-score values of the LSTM on the training set per fold are quite similar and tend to be stable. On the validation set, the F1-score curve essentially follows the same fluctuation trend as that of the training set, and the fluctuation range is reduced to varying degrees, with the variance decreasing, indicating that both hyperparameter optimization methods reduced the overfitting risk of the LSTM.

By combining the quantitative analysis in Table 2 and the qualitative analysis in Figure 17, it can be concluded that both hyperparameter optimization methods enhanced the generalization ability of the LSTM and lowered its risk of overfitting.

After the above performance analysis and overfitting risk analysis of the optimized models, we can draw the following conclusion: the LSTM model after chaotic optimization is the best model among those used in this experiment. On the dataset used in this experiment, this model has a controllable overfitting risk and demonstrates excellent and reliable classification ability for the sound samples of pneumonia patients and healthy individuals.

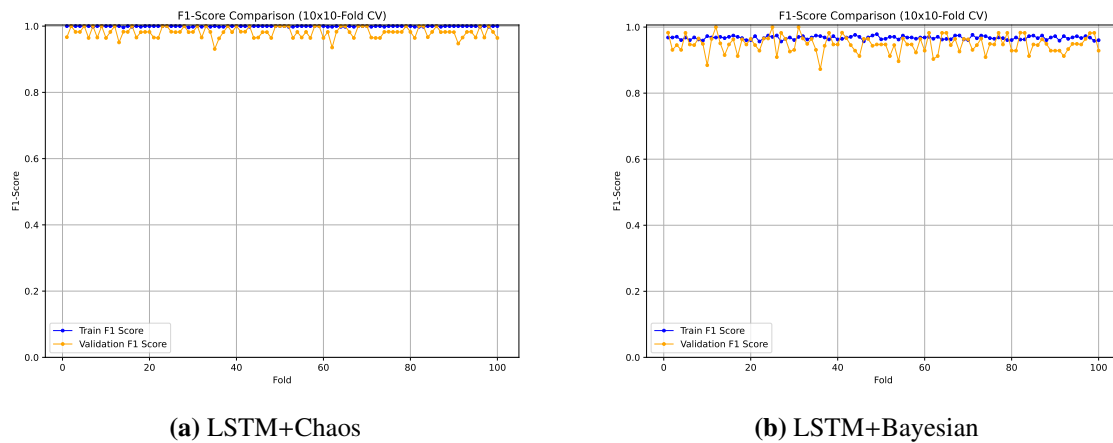


Figure 17. Comparison of ROC curves for the LSTM after different optimizations.

6. Conclusions

This experiment compared the performance differences of various machine learning models on the selected audio dataset. When extracting features from the audio in the dataset, we chose to use MFCC features. For each audio file, 13 MFCC means and 13 MFCC standard deviations were extracted, along with 4 time-domain features, which were then combined to form a 30-dimensional feature vector corresponding to the audio file. Subsequently, machine learning models such as the BP and the RNN were employed to perform binary classification training and evaluation on these feature vectors. Ten repetitions of ten-fold cross-validation were used during training to more comprehensively demonstrate the models' generalization ability on the dataset. Based on the cross-validation, the models' performance evaluation metrics, including accuracy, precision, recall, F1-score, as well as confusion matrices, ROC curves, and F1-score comparison charts, were calculated.

The results showed that the LSTM model after chaotic optimization exhibited a significant classification capability on the selected pneumonia dataset, and the optimal hyperparameters obtained by chaotic optimization were as follows: learning rate (lr) = 1.17×10^{-3} , hidden layer dimension ($hidden_dim$) = 17, early stopping patience ($patience$) = 5, and batch size ($batch_size$) = 21. After ten repetitions of ten-fold cross-validation, the model achieved an accuracy of 97.49%, a precision of 98.21%, a recall of 98.18%, and an F1-score of 98.17% on the validation set, indicating that the model was able to correctly predict the vast majority of both pneumonia patients' audio and healthy individuals' audio in this dataset, demonstrating excellent classification performance. Meanwhile, the model's risk of overfitting was controllable, with a maximum metric difference of only 2.43% between the training and validation sets, suggesting good generalization ability across different data splits.

This study has certain limitations. In the study, machine learning models were used for binary classification of pneumonia patients and healthy individuals, and the selected dataset only contained 418 audio samples. For the prediction of a major and hazardous disease such as pneumonia, the dataset required for model training must be extensive and representative. Therefore, the models in this experiment may only be applicable to the dataset used, and the breadth, representativeness, and applicability of the dataset cannot be guaranteed. Additionally, the variety of models and

hyperparameter optimization methods involved in the training was insufficient, and there may exist other models more suitable for this classification task.

In the future, to address the limitations of this study, we plan to make further improvements. For instance, we will expand the scale of the pneumonia audio dataset and seek larger pneumonia audio sample datasets to make the experimental results more accurate and authoritative. We will explore other sound features more suitable for distinguishing between pneumonia patients and healthy individuals to enhance the model's ability to discriminate pneumonia patients' sound samples and improve model performance. We will consider using other machine learning or deep learning models, or ensemble models, such as combining the RNN and the LSTM, which may yield better classification performance on the pneumonia dataset.

Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by College Students' Innovative Entrepreneurial Training Plan Program under Grant No. 202410459130.

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. Asif MAAR, Nishat MM, Faisal F, Dip RR, Udoy MH, Shikder MF, et al. (2021) Performance evaluation and comparative analysis of different machine learning algorithms in predicting cardiovascular disease. *Eng Lett* 29: 731.
2. Gomathy CK, Naidu MAR, (2021) The prediction of disease using machine learning. *Int J Sci Res Eng Manage* 5: 1–7.
3. Parthiban G, Srivatsa SK, (2012) Applying machine learning methods in diagnosing heart disease for diabetic patients. *Int J Appl Inf Syst* 3: 25–30. <https://doi.org/10.5120/ijais12-450593>
4. Patel J, Tejal Upadhyay D, Patel S, (2015) Heart disease prediction using machine learning and data mining technique. *Heart Dis* 7: 129–137.
5. Shahbaz M, Ali S, Guergachi A, Niazi A, Umer A, (2019) Classification of Alzheimer's disease using machine learning techniques. *Data* 2019: 296–303. <https://doi.org/10.5220/0007949902960303>
6. Nishat MM, Faisal F, Dip RR, Nasrullah SM, Ahsan R, Shikder F, et al. (2021) A comprehensive analysis on detecting chronic kidney disease by employing machine learning algorithms. *EAI Endorsed Trans Pervasive Heal Technol* 7: e1. <https://doi.org/10.4108/eai.13-8-2021.170671>

7. Shukur BS, Mijwil MM, (2023) Involving machine learning techniques in heart disease diagnosis: A performance analysis. *Int J Electr Comput Eng* 13: 2177. <https://doi.org/10.11591/ijece.v13i2.pp2177-2185>
8. Pingale K, Surwase S, Kulkarni V, Sarage S, Karve A, (2019) Disease prediction using machine learning. *Int Res J Eng Technol* 6: 831–833.
9. Tawfik M, Al-Zidi N M, Fathail I, Nimbhore S, (2022) Asthma detection system: Machine and deep learning-based techniques. In: *Artificial Intelligence and Sustainable Computing: Proceedings of ICSISCET 2021*, 207–218. https://doi.org/10.1007/978-981-19-1653-3_16
10. Mafazy MM, Faisal MR, Kartini D, Indriani F, Saragih TH, (2023) Classification of covid-19 cough sounds using mel frequency cepstral coefficient (MFCC) feature extraction and support vector machine. *Telematika* 16: 60–68. <http://dx.doi.org/10.35671/telematika.v16i2.2569>
11. Suliman A, Zhang Y, (2015) A review on back-propagation neural networks in the application of remote sensing image classification. *J Earth Sci Eng* 5: 52–65. <https://doi.org/10.17265/2159-581X/2015.01.004>
12. Sibi P, Jones SA, Siddarth P, (2013) Analysis of different activation functions using backpropagation neural networks. *J Theor Appl Inf Technol* 47: 1264–1268.
13. Zhang XR, Sun X, Sun W, Xu T, Wang PP, Jha SK, (2022) Deformation expression of soft tissue based on BP neural network. *Intell Autom Soft Comput* 32: 1041–1053. <https://doi.org/10.32604/iasc.2022.016543>
14. Xi WD, Huang L, Wang CD, Zheng YY, Lai J, (2019) BPAM: Recommendation based on BP neural network with attention mechanism. *IJCAI* 2019: 3905–3911. <https://doi.org/10.24963/ijcai.2019/542>
15. Zhou H, Chen Y, Zhang S, (2019) Ship trajectory prediction based on BP neural network. *J Artif Intell* 1: 29. <https://doi.org/10.32604/jai.2019.05939>
16. Gunarto DM, Sa'adah S, Utama DQ, (2023) Predicting cryptocurrency price using RNN and LSTM method. *Sisfokom* 12: 1–8.
17. Smailov N, Dosbayev Z, Omarov N, Sadykova B, Zhekambayeva M, Zhamangarin D, et al. (2023) A novel deep CNN-RNN approach for real-time impulsive sound detection to detect dangerous events. *Int J Adv Comput Sci Appl* 2023: 14. <https://doi.org/10.14569/IJACSA.2023.0140431>
18. Krak I, Zalutka O, Molchanova M, Mazurets O, Bahrii R, Sobko O, et al. (2024) Abusive speech detection method for Ukrainian language used recurrent neural network. *COLINS* 2024: 16–28. <https://doi.org/10.31110/COLINS/2024-3/002>
19. Mahdi M G, Sleem A, Elhenawy I, Safwat S, (2024) Advancing Arabic handwriting recognition with convolutional and recurrent neural network ensembles. *Inf Sci Appl* 3: 43–63. <https://doi.org/10.61356/j.iswa.2024.3294>
20. Adegbenle A, Awodele O, Ogbonna C, Adigun T, (2024) A secured keystroke-based model for preventing social engineering attacks using recurrent neural network. *Int J Comput Sci* 13: 1–11. <https://doi.org/10.47760/ijcsmc.2024.v13i10.001>

21. Al-Selwi SM, Hassan MF, Abdulkadir SJ, Muneer A, (2023) LSTM inefficiency in long-term dependencies regression problems. *J Adv Res Appl Sci Eng Technol* 30: 16–31. <https://doi.org/10.37934/araset.30.3.1631>
22. Albeladi K, Zafar B, Mueen A, (2023) Time series forecasting using LSTM and ARIMA. *Int J Adv Comput Sci Appl* 14: 313–320. <https://doi.org/10.14569/IJACSA.2023.0140133>
23. Issa ASA, Albayrak Z, (2023) DDoS attack intrusion detection system based on hybridization of CNN and LSTM. *Acta Polytech Hung* 20: 105–123. <https://doi.org/10.12700/APH.20.2.2023.2.6>
24. Klosowski G, Rymarczyk T, Niderla K, Kulisz M, Skowron Ł, Soleimani M, (2023) Using an LSTM network to monitor industrial reactors using electrical capacitance and impedance tomography-a hybrid approach. *Eksploat Niezawodn* 25. <http://doi.org/10.17531/ein.2023.1.11>
25. Lasri I, Riadhsoh A, Elbelkacemi M, (2023) Self-attention-based Bi-LSTM model for sentiment analysis on tweets about distance learning in higher education. *Int J Emerg Technol Learn* 18: 119–141. <https://doi.org/10.3991/ijet.v18i12.38071>
26. Adam AZR, Setiawan EB, (2023) Social media sentiment analysis using convolutional neural network (CNN) dan gated recurrent unit (GRU). *JITEKI* 9: 119–131. <https://doi.org/10.26555/jiteki.v9i1.25813>
27. Darmawan H, Yuliana M, Hadi M, (2023) GRU and XGBoost performance with hyperparameter tuning using GridSearchCV and Bayesian optimization on an IoT-based weather prediction system. *Int J Adv Sci Eng Inf Technol* 13: 848–859. <https://doi.org/10.18517/ijaseit.13.3.18377>
28. Dey A, Biswas S, (2024) Workout action recognition in video streams using an attention driven residual DC-GRU network. *Comput Mater Continua*, 79: 3067–3087. <https://doi.org/10.32604/cmc.2024.049512>
29. Kervanci IS, Akay MF, Özceylan E, (2024) Bitcoin price prediction using LSTM, GRU and hybrid LSTM-GRU with bayesian optimization, random search, and grid search for the next days. *J Ind Manage Optim*, 20: 570–588. <https://doi.org/10.3934/jimo.2023091>
30. Kamil G, Setiawan EB, (2023) Aspect-level sentiment analysis on social media using gated recurrent unit (GRU). *Build Inf Technol Sci* 4: 1837–1844. <https://doi.org/10.47065/bits.v4i4.3105>
31. Sun G, Lin J, Yang C, Yin X, Li Z, Guo P, et al. (2021) Stock price forecasting: An echo state network approach. *Comput Syst Sci Eng* 36: 509–520. <https://doi.org/10.32604/csse.2021.014189>
32. Lukoševicius M, (2007) Echo state networks with trained feedbacks. *Networks* 2007: 4.
33. Hart AG, Olding KR, Cox AMG, Isupova O, Dawes JHP, (2021) Echo state networks for reinforcement learning, preprin, arXiv:2102.06258.
34. Illies I, Jaeger H, Kosuchinas O, M Rincon, V Sakenas, N Vaskevicius, *Stepping Forward through Echoes of the Past: Forecasting with Echo State Networks. Short Report on the Winning Entry to the NN3 Financial Forecasting Competition*, 2007. Available from: https://www.ai.rug.nl/minds/uploads/2213_Jaegeretal07b.pdf.
35. Saadat J, Moallem P, Koofigar H, (2017) Training echo state neural network using Harmony search algorithm. *Int J Artif Intell* 15: 163–179.

-
36. Zhang H, Ren G, (2024) Intelligent leaf disease diagnosis: Image algorithms using Swin Transformer and federated learning. *Visual Comput*, 41: 4815–1838. <https://doi.org/10.1007/s00371-024-03692-w>



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)