Computers and Electrical Engineering 92 (2021) 107156

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

IncLSTM: Incremental Ensemble LSTM Model towards Time Series Data

Huiju Wang, Mengxuan Li, Xiao Yue *

School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China

ARTICLE INFO

Keywords:
Incremental learning
Time series data
IncLSTM
FL-Share
Ensemble learning
LSTM neural network
Transfer learning

ABSTRACT

Long short-term memory (LSTM) is one of the most widely used recurrent neural network. Traditionally, it adopts an offline batch mode for model training. To be updated with new data, the network has to be re-trained with merged data using both old and new data, which is very time-consuming and causes catastrophic forgetting. To address this issue, we proposed an incremental ensemble LSTM model-IncLSTM, which fuses ensemble learning and transfer learning to implement incremental updating of the model. The experimental results showed that, in average, the proposed method decreases training time by 18.8%, and improves the prediction accuracy by 15.6% compared with the traditional methods. More importantly, the larger the training data size is, the more efficient IncLSTM would be. While updating the new model, current model predicts independently and concurrently, and the switch between current model and new model occurs once the update is completed, which significantly improves the training efficiency of the model.

1. Introduction

Long short-term memory (LSTM) recurrent neural network, one of the most widely used recurrent structures in sequential data modeling, can model complex nonlinear time series data due to its strong nonlinear fitting ability and memory characteristic [1]. Up to the date, more than 1000 studies has devoted to LSTM and its variants, and has contributed to the development of well-known projects such as Google Translate, Google voice assistant, Alexa, Siri, Cortana, etc [2].

Traditionally, LSTM models are trained in one offline batch way, and all the training data are input into the learning model at one time. Once trained, the neural network model is fixed and difficult to be adjusted. Thus, it is difficult for traditional LSTM models to train data with rapid dynamic changes in reality, such as thermal power plant data, weather data, and stock market data. In order to fully mine the information in the new data to update the model, we need to continuously merge the old data with the new data and re-train. However, this requires a time-consuming process and causes the problem of catastrophic forgetting [3]. Also, the time series data is challenging to be accurately trained with such an approach.

This work is supported by the Central Government Special Financial Foundation for International Exchanges and Cooperation Project of Zhongnan University of Economics and Law, Zhongnan University of Economics and Law Teaching and Research Project under Grant no. YB202068, "the Fundamental Research Funds for the Central Universities", Zhongnan University of Economics and Law under Grant no. 2722020PY047, Natural Science Foundation of Hubei Province under Grant no. 2017CFB592.This paper is for regular issues of CAEE. Reviews processed and approved for publication by the co-Editor-in-Chief Huimin Lu.

* Corresponding author.
E-mail address: yuexiao@zuel.edu.cn (X. Yue).

https://doi.org/10.1016/j.compeleceng.2021.107156
Received 16 January 2021; Received in revised form 6 March 2021; Accepted 6 April 2021
Available online 25 April 2021

Incremental learning continually learns new knowledge from new training data while preserving most of the features learned before. Different from the offline batch learning, incremental learning is also able to update model parameters in a sequential data stream incrementally. The incremental learning models often have better scalability and lower memory requirements since incrementally updating a network model consumes a short time than a fully re-training approach.

In this work, we attempt to apply the idea of incremental learning and ensemble learning on the LSTM model, so as to make it have the characteristics of dynamic evolution for the prediction of time-series data. Inspired by Bagging ensemble learning, we proposes an incremental ensemble LSTM model-IncLSTM, which dynamically updates the model according to the following time series data for accurate real-time prediction. In the method, arriving real-time data stream is sliced into several sub-datasets at one-time interval, each new generated sub-dataset is trained independently based on transfer learning and produced one weak learner. The new generated weak learners and the old ones are integrated to generate a strong learner. The update and the forecasting of deployed model run independently and concurrently, and the switch between updated model and deployed model occurs once model update has completed. We incorporated a series of optimization techniques, including FL-Share algorithm, sample transfer learning, weak learners buffer algorithm, etc, to reduce the training time further. The experiments on real time series dataset verify that IncLSTM outperforms counterparts both on the performance and accuracy in most cases, and bigger size of training data is, more obvious performance advantage IncLSTM would be. In average, IncLSTM decreases training time by 18.8%, reduces the loss index by 10.13% and improves the model fitting degree by 0.51% compared with the traditional methods.

Our contributions may be summarized as follows:

(1) proposed a novel variant of LSTM – IncLSTM, which may train LSTM model incrementally in one scalable way by combining ensemble learning and transfer learning. To the best of our knowledge, this is the first effort on incremental training of LSTM model.
(2) incorporated a series of optimization algorithms for IncLSTM. including FL-Share transfer learning algorithm, sample-based transfer learning, weak learner buffer algorithm, which improve training further efficiently.
(3) conducted extensive experiments and verified the efficiency of IncLSTM.

2. Related work

According to the type of base model, incremental time series model may be divided into two main types: one is traditional time series models based incremental model, and the other one is deep learning based incremental model. As the former one, such as ARMA-ONS [4], is designed towards data that follows linear relationship and is difficult to model nonlinear data, we discuss mainly on latter type here.

For deep learning based time series model, incremental learning may be divided into three types: self-organizing incremental neural network, structural adaptation based incremental neural network and ensemble learning based incremental neural network. As our work belongs to the last one, we mainly review related ensemble learning work here. The incremental ensemble learning algorithm usually learns by weighting several different basic models, which preserves the previously learned historical knowledge and solves the data distribution changes.

2.1. Ensemble-based incremental neural network

In Learn++ algorithm, several learners are integrated using voting weight for the multi-classification problem, in which a weak learner with high classification performance receives a higher voting weight [5]. Shen et al. proposed an improved fast Learn++ NSE algorithm to optimize the weight calculation process by using the sliding window technology [6]. Gangardiwala et al. proposed a dynamically updated voting weight of the classifier based on the position of test input in the feature space to improve the Learn ++ incremental learning algorithm [7]. This method was proved to have a stronger immunity to “catastrophic forgetting” and to be more stable and malleable. Learn# designed one incremental learning structure for text classification based on reinforcement learning and XGBoost [8]. Xiang et al, proposed a dynamic correction vector based incremental learning algorithm to address both the bias problem from knowledge distillation and the overfitting problem [9]. However, most existing incremental learning algorithms use online convex optimization to learn the shallow model, such as the linear method [10] or kernel method [11], which cause such methods cannot learn complex nonlinear functions required often in practical application scenarios [12]. And such existing works are based on the idea of Adaboost and belongs to the Boosting algorithm, which has a strong dependency between each weak learner and each weak learner has to be generated serially and is difficult to meet the potential needs of parallel training. Differently, we designed incremental LSTM based on Bagging algorithm which may train new dataset in parallel to improve the training performance. In addition, we utilized the characteristics of adjacent datasets which are ignored by Learn++ like algorithms, to improve prediction accuracy.

2.2. Ensemble neural network

In recent years, several studies adopted the idea of ensemble learning algorithm into the neural network. Sarwar et al. proposed incremental growing deep convolutional neural network based on model transfer learning [13]. Mosca et al. proposed a deep incremental boosting to reduce the time consumption of training and improve generalization [14]. Katuwal et al. proposed a deep RVFL (Random vector functional link) network (dRVFL) with stacked layers and an integrated deep network (edRVFL) [15]. Compared to traditional integration methods that require independent training of multiple models from scratch, edRVFL is obtained by training a single dRVFL network at one time. Li et al. proposed a novel incremental semi-supervised learning framework for streaming data classification [16]. Most existing works of the incremental ensemble neural network mainly directly add hidden layer neurons to update the model or add the neural network to the existing neural network to integrate, and these two update methods will both lead to the increasing of the network complexity and the training scale, which cause the sharp decrease of the training speed and generalization ability of the model. To avoid these, our proposal adopts stable network structure and updates model incrementally by adjusting the weights.

To summarize, most existing incremental ensemble neural network algorithms focused on clustering and classification algorithms, and rarely on prediction which is our main concern. Besides, to the best of our knowledge, little work has gone to incremental training of LSTM model.

3. Design of IncLSTM

In this part, we discuss key designs of IncLSTM. Based on the characteristics of time series data and neural networks, we assume the followings:

Assumption 1: The data in the next times are highly correlated in the data stream. Thus, the data between the current time and the previous time are related to each other. Also, the datasets generated in adjacent periods are similar.[1]

Assumption 2: As time goes by, the amount of data becomes more massive, and the data distribution changes gradually. In other words, the feature space of adjacent datasets is the same, while the marginal probability changes gradually [17,18].

Assumption 3: To train the model takes much longer than to generate the data, and input data are continuously generated. The model prediction is performed in real-time [19].

3.1. Framework of IncLSTM

The overall flow chart of the IncLSTM model is shown in Fig. 1. Firstly, data stream is sliced by time T so that a sub-dataset is generated every time T. K sub-datasets are generated in chronological order, namely D_k, k={1, 2, …, K}. It is possible that in some cases, time series data stream may be randomly interrupted occasionally. To alleviate this, we use a data buffer pool to store arrived time series data. Every time T, the data in the buffer will be used for model training and then be emptied after training.

Inspired by the Bagging idea in ensemble learning [20], weak learners $Learner_k$, k={1, 2, …, K} are generated by parallel training for K sub-datasets. All weak learners are multi-layer bidirectional LSTM neural network models. The transfer learning is used to achieve a better prediction effect on a small dataset with a small parameter update cost. The weak learners are weighted in the order of training completion, and thus the strong learner can predict the time series data. As time goes by, the newly generated weak learner incrementally updates the strong learner with the change of datasets. IncLSTM model preserves its performance over the learned dataset (avoiding catastrophic forgetting) while learning new knowledge.

3.2. Weak learners construction

Bidirectional LSTM is chosen as the backbone model based on its following advantages [21,22]: 1) It can avoid the gradient disappearance and gradient explosion, which is easily led by the traditional RNN in the long time sequence. 2) It can learn time-dependent information. 3) It can take advantage of valid information from both the past and the future.

Multiple bidirectional LSTM layers are superimposed to implement the deep feature mining of time series data through the multi-layer neural network structure. Finally, the deep bidirectional LSTM network (Deep-bi-LSTM) is formed, which is used as the initial training model of a weak learner. The model structure is depicted in Fig. 2.

3.3. Strong learner ensemble

We use the Bagging algorithm to integrate several weak learners. The Bagging algorithm allows to train multiple weak learners independently and in parallel, combines them together according to specific rules. Since the training time of the neural network is much longer than the time of data generation and model prediction, the Bagging algorithm, which supports parallel learning, can be applied to incremental neural networks effectively. Aiming at the continuously generated time-series data, the new dataset may be trained in parallel. The generated weak learners are quickly combined through an ensemble algorithm, and the updated strong learner

[1] https://www.cnblogs.com/tjpeng/p/12368665.html

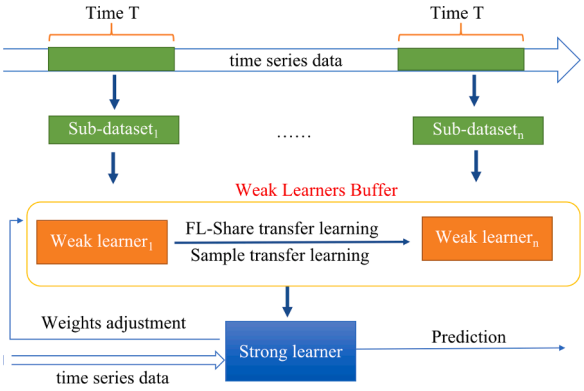


Fig. 1. The flow chart of IncLSTM.

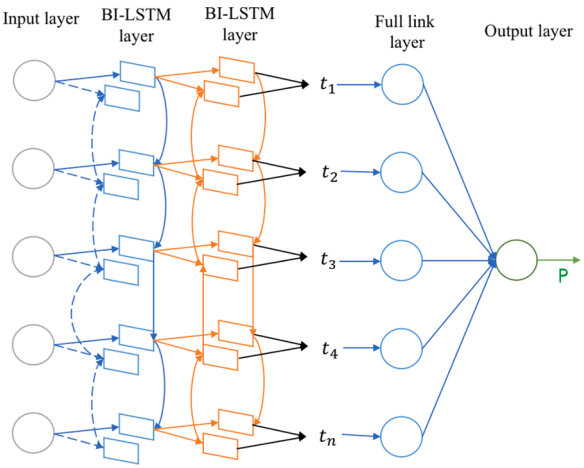


Fig. 2. The model structure of weak learner.

is used to conduct the predict operation.

The $n{+}1^{th}$ weak learner $Learner_{n+1}$ is trained using the previous weak learners $Learner_1$, $Learner_2$, …, $Learner_n$. The strong learner, $StrongLearner_n$, is updated by adjusting the weighting coefficients of each weak learner on the dataset D_{n+1}. The error loss e_i between the predicted value and the ground-truth value of each weak learner is computed by using the error formula used in Adaboost R^2 regression algorithm. The detailed steps of the strong learner generation are described in Algorithm 1.

4. Optimization of incremental training

In this section, we discuss our optimization techniques, including VeryFastKMM-Tradaboost based sample transfer learning, FL-Share transfer learning and buffer pool algorithm. The sample-based transfer learning reduces the difference in data distribution between adjacent datasets and utilizes useful information from old datasets for data expansion [23]. The FL-Share transfer learning makes the model adapt to the characteristics of the new dataset and improve its prediction accuracy. The buffer pool strategy avoids the infinite growth of the number of weak learners, maintaining the stability of the model.

4.1. VeryFastKMM-Tradaboost based sample transfer learning

Training a neural network is usually a very time-consuming process. Adjacent datasets are correlated in the stream-data (Assumption 1). Accordingly, the previously trained model in the previous time-period can be used as a pre-trained model to learn

new knowledge instead of from scratch. Utilizing the knowledge obtained from the previous time-period accelerates and optimizes the learning process. However, the data distribution in stream-data is gradually changed over time, so the marginal probability distribution between adjacent datasets are not the same (Assumption 2). Based on Assumptions 1 and 2, the sample-based transfer learning method is employed In this work.

According to specific weight generation rules, the training data samples having different distribution are reused to mine the useful information from the old dataset fully. The new weak learner model is trained with reused old data and new datasets.

Tradaboost model achieves excellent results when the source and target fields are similar to each other while solving the problem of different data distributions between the training dataset and the test dataset effectively [24]. In this work, the Tradaboost model is introduced into the transfer learning of weak learners to extract useful samples conforming to the distribution of the new dataset from the previous historical dataset.

4.1.1. Problem statement

In the time-series of data, each sub-dataset is generated according to the order, forming historical dataset D_{old} and the target dataset D_{new} (new datasets). Assuming that a weak learner $Learner_{old}$ has been generated by training on the historical dataset D_{old}, we can divide the target dataset D_{new} into the training set X_{new} and the test set S. The entire training dataset T is defined as $T \in \{(X_{old} \cup Xnew) * Y\}$, where Y is the ground-truth set for all training data.

The entire training data T is divided into two datasets:

$Told = \{(x_i^{old}, y(x_i^{old})\}$, where $x_i^{old} \in X_{old}$ and i=1,2,…,n.

$Tnew = \{(x_j^{new}, y(x_j^{new})\}$, where $x_j^{new} \in X_{new}$ and j=1,2,…,n. where $y(x)$is the ground-truth value of x. T_{new} and test data S have the same distribution, while T_{new} and test data S have different distributions. A new weak learner $Learner_{new}$ is trained on the entire training dataset, T.

4.1.2. Weights initialization in Tradaboost

According to the weight distribution strategy of the original Tradaboost algorithm, the samples in the auxiliary data set will get the same initial weight, and such initial weights will be adjusted in the subsequent iteration process according to the difference between the predicted value and the true value of each sample. However, due to the big distribution difference between samples in the auxiliary data set and the target data set, this weight distribution strategy is unreasonable. In order to solve this problem, an improved KMM algorithm VeryFastKMM [25] are adopted to optimize Tradaboost's weight initialization strategy: auxiliary data samples with higher similarity to the target data set are given higher initial weights, and samples with lower similarities are given a lower weight. According to [26], weights are initialized in Tradaboost transfer algorithm as follows:

$$w = \begin{cases} \beta = \text{VeryFastKMM}(T_a, T_b), sampleweightsofhistoricaldataset \\ \max(\beta), sampleweightsoftargetdataset \end{cases} \tag{1}$$

4.1.3. Tradaboost error function

The Tradaboost algorithm is originally designed for the classification where each sample x_i is 0 or 1, and thus the error is $e_i \in \{0, 1\}$. On the contrary, in the regression problem, the error e_i is likely to be greater than 1, causing the weight update coefficient β<0 when Tradaboost algorithm updates the weight of the sample. Therefore, the error function in Adaboost R^2 is employed, and the maximum error on the training set T was calculated as follows:

$$E = \max|y_i - f_i(x_i)|, i = 1, 2, 3, \ldots, n + m. \tag{2}$$

Then, the relative error e_i for each sample is calculated as follows:

$$e_i = \begin{cases} \dfrac{y_i - f_i(x_i)}{E}, linearerror \\ \dfrac{(y_i - f_i(x_i))^2}{E^2}, squareerror \\ 1 - \exp\left(-\dfrac{|y_i - f_i(x_i)|}{E}\right), indexerror \end{cases} \tag{3}$$

Here, the exponential error is used to measure the relative error of the samples.

4.1.4. Sample transfer learning

Sample based Tradaboost algorithm is adopted for transfer learning between weak learners. The VeryFastKMM algorithm is used to initialize the weights of the historical sample sets. A regression error function used in Adaboost R^2 is adopted to calculate the relative sample error. The detailed steps of the sample transfer algorithm are described in Algorithm 2.

If the prediction error on the historical training dataset is higher than that of the target training dataset, data distribution of target dataset should be different from historical dataset. To reduce the weight of data having different distribution, the data is multiplied by $\rho^{|y_i - f_t(x_i)|}$. Because the value of ρ is between 0 and 1, the impact of samples with higher prediction errors on the new weak learner is reduced in the next iteration. After several iterations, the data in the historical dataset that conforms to the distribution of the target dataset should have a higher weight. In contrast, the sample weight that does not conform to the distribution of the target data will be reduced. On the contrary, a high prediction error in the target training dataset means that the existing weak learner model is not fitted the target dataset. Thus, the weight of the target sample data is increased to reduce the prediction error in the next training, to make the weak learner model better fit the target data set. The sample having the adjustment weight less than 0.5 is removed to improve the training efficiency.

4.2. FL-Share transfer learning algorithm

4.2.1. Fine-tuning

Two adjacent sub-datasets can share some model parameters since the difference of temporally adjacent data is small in stream data (Assumption 1). Once the original dataset has been trained, a new neural network can be trained from the pre-trained model avoiding the time-consuming re-training process.

Thus, the model transfer learning is used to train weak learners for datasets in multiple sequences. The model parameters of the weak learner $Learner_{t-1}$ from the previous dataset are used as initial parameters for layers of the new weak learner model. Then, the new weak learner $Learner_{t-1}$ is fine-tuned with a new dataset D_t. The Fine-tuning diagram is illustrated in Fig. 3.

4.2.2. Shared layer

To reuse the weights of the existing network achieves better performance than a simple fine-tuning transfer learning and learning from scratch. Inspired by the idea of controller module in the DAN model [27], we propose the concept of the shared layer to realize further model migration between weak learners, of which algorithm is layer and combined with some weighted ratio, to extract the information from the original neural network and combine it with current information (as illustrated in Fig. 4).

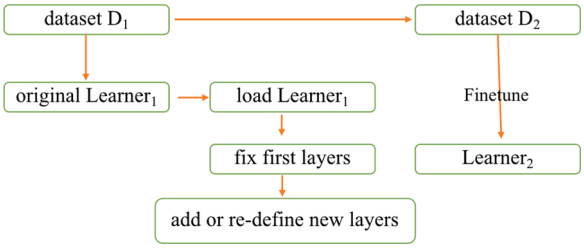


Fig. 3. The Fine-tuning diagram of the weak learner.

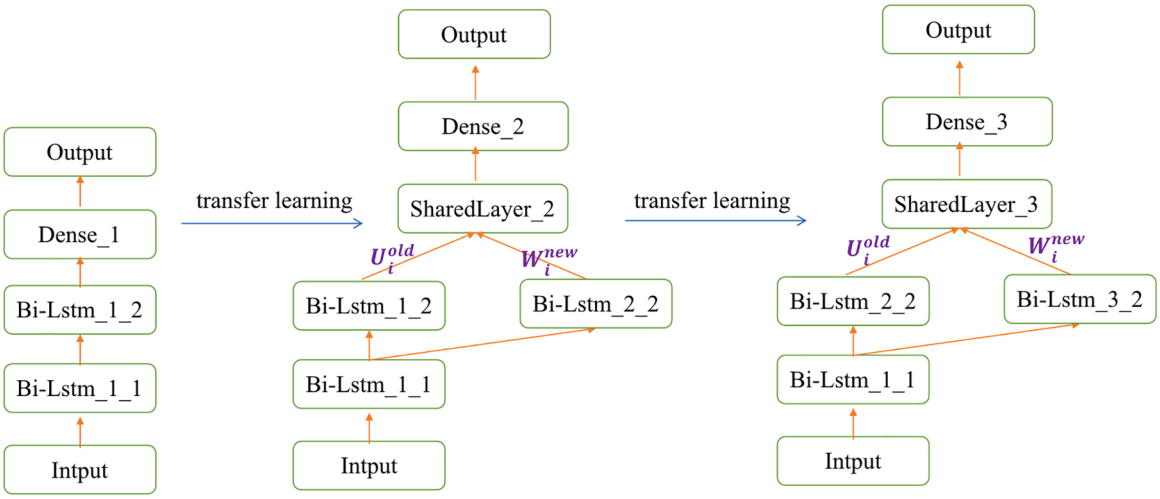


Fig. 4. Layer sharing based transfer learning of weak learner.

Table 1
Time series dataset information.

No	Attribute	Data Type	Description	No	Attribute	Data Type	Description
1	Year	int	year of data in this row	11	CO	float	CO concentration (ug/m^3)
2	Month	int	month of data in this row	12	SO_2	float	SO_2 concentration (ug/m^3)
3	Day	int	day of data in this row	13	TEMP	float	temperature(degree Celsius)
4	Hour	int	hour of data in this row	14	PRES	float	pressure (hPa)
5	PM2.5	float	PM2.5 concentration (ug/m^3)	15	DEWP	float	dew point temperature (degree Celsius)
6	PM10	float	PM10 concentration (ug/m^3)	16	TEMP	float	temperature(degree Celsius)
7	SO_2	float	SO_2 concentration (ug/m^3)	17	RAIN	float	precipitation (mm)
8	CO	float	CO concentration (ug/m^3)	18	wd	float	wind direction
9	O_3	float	O_3 concentration (ug/m^3)	19	WSPM	float	wind speed (m/s)
10	NO_2	float	NO_2 concentration (ug/m^3)	20	Station	char	name of the air-quality monitoring site

4.2.3. FL-Share transfer learning

Given two adjacent datasets D_{t-1} and D_t, $Learner_{t-}$ is trained with the current dataset D_t. Note that $Learner_{t-1}$ has been learned with D_{t-1} and $Learner_t$ has the same architecture as $Learner_{t-1}$. The extension conducts based on original $Learner_{t-1}$, that is, based on the last layer Bi-LSTM of the $Learner_{t-1}$. Meanwhile, a new Bi-LSTM layer is trained. Furthermore, the two layers are connected to the shared layer. The outputs of the original layer and the new layer are weighted by the weight matrix U and W. In other words, the input of the shared layer is the fusion of the output information of the historical neural network layer and the output information of the current neural network layer.

Notably, the shared layer (Share_Layer) does not change the output dimension of the data, but only weights the output of the bi-LSTM layer in the original network with the output of the newly trained bi-LSTM layer.

$$h_i^{(sharedLayer)} = f\left(U_i^{old} * h_{i-1}{}^{(old)} + W_i^{new} * h_{i-1}{}^{(new)}\right), \tag{4}$$

where $(x)=max(0,x)*U_i^{old}$ is the weight matrix for adjusting the output of the original bi-LSTM layer and W_i^{new} is the weight matrix for adjusting the output of the new bi-LSTM layer. When the weight matrix $U_i^{old,}$ W_i^{new} are initialized, the constants α and β are used for initialization with $\alpha + \beta = 1$. Weight matrices U_i^{old} *and* W_i^{new} are added to the training parameters of the overall neural network. The FL-Share transfer learning for weak learners is shown in Algorithm 3.

4.4. Weak learners buffer pool

The strong learner is a weighted combination of all the weak learners. If the number of weak learners is not limited, the weak learners will be added to the updated list of the strong learners indefinitely, resulting in the continuous increase of the time and space costs predicted by the strong learners. Also, the increased number of weak learners causes memory overhead in the operating system.

The buffer pool ρ of a weak learner is used to limit the maximum number of weak learners. The most recent ρ weak learners are stored in the buffer pool. Further, to store weak learners adaptively, the prediction error tolerance coefficient (γ) is employed. When the difference of the prediction error of the current weak learner and the maximum error in the buffer pool is less than γ, the current weak learner is stored in the buffer pool, while the weak leaner with the maximum error is removed. The buffer pool algorithm is shown in Algorithm 4.

5. Experimental results

5.1. Dataset and experimental environment

In this work, the Beijing multi-site air quality dataset [28] published on the UCI machine learning library was used as the experimental data. The dataset contains a total of 420,768 pieces of data. The data information is shown in Table 1. PM 2.5 concentration is selected as the primary indicator of air pollution, and the PM2.5 index of the next hour is predicted based on the air quality data of the previous few hours. All the algorithms are implemented in Python. The experiments are conducted on one server with Intel Core (TM) i5-3230M CPU (2.60 GHz), 12 GB main-memory, and 300GB SSD+500 GB HDD Disk.

5.1.1. Model evaluation index

Five indices are used to evaluate the performance of a regression model: Explained variance score (EV), mean absolute error (MAE), mean square error (MSE), root mean square difference (RMSE), and the determination coefficient (R^2).

5.1.2. Experiment details and implementation

Three programs were developed to implement the proposed algorithm model: time series simulation emitter, model incremental training and update, and real-time data prediction. In the data emitter, the preprocessed dataset was divided into N sub-datasets and sent to the incremental training and updating program to get initial model; after such initialization, input data are accumulated and sent to the incremental training and updating program at each time interval T. In the model training and updating program, the process pool was opened to simulate training of sub-datasets. Multiple weak learners trained were sent to the strong learner tuning function

through the communication queue in sequence. The newly generated dataset D_{n+1} was used to adjust the weights of the first n weak learners that had been trained to form the latest integrated strong learner $StrongLearner_n$. In the process of a single weak learning training, each sub-dataset D_n was divided into the training data and test data in a ratio of 7:3.

The integrated LSTM neural network model was implemented using Keras. The key parameters in the experiment are as follows: the number_of_dataset=6, activation=Relu, drop_out=0.1, tradaboost_number=2, optimizer= Adaptive Moment Estimation, and epochs=50.

The whole dataset was divided into six sub-datasets, namely $Data_1$, $Data_2$, ..., and $Data_6$, and the sub-datasets were merged to train the model and test the model, simulating the growth and prediction of time series data in reality. For convenience of discussion, we have $Group_i = \bigcup_{i=1}^{m} Data_i$.

5.1.3. Baseline methods

In order to verify the feasibility of the incremental ensemble LSTM neural network model in the prediction of time series data, several comparisons were conducted.

This approach of online LSTM model toward time-series data would be novel in model training, allowing LSTM model fit the input data adaptively on time, which is is a new modeling way for time series data. Thus, the original BiLSTM and ARMA-ONS [4] were compared to verify the performance of IncLSTM in incremental updates. ARMA-ONS is the online sequential model based on traditional methods. ADF test was used to detect the stationarity of data. Further, a white noise test was conducted on the original data.

The proposed weak learner training schemes were compared to other methods used in the prediction process of stream-data. The compared methods are as follows:

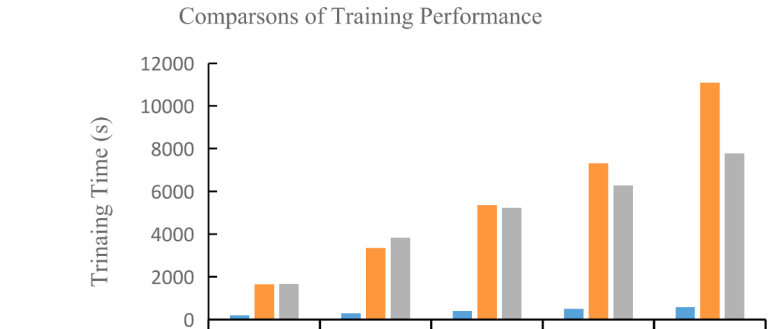
1) Standard-BiLSTM: the BiLSTM model with Fine-tuning is compared in training each sub-dataset.
2) Instance-based-Transfer-BiLSTM: the VeryFastKMM-Tradaboost sample transfer algorithm with BiLSTM was compared in training each sub-dataset.
3) IncLSTM: BiLSTM model using the VeryFastKMM-Tradaboost sample transfer algorithm and FL-share transfer algorithm based on Fine-tuning and shared layer were compared. The whole model proposed in this work was used to train each sub-dataset to compare the improve effect of the sharing layer mechanism in the model.

5.2. Model training efficiency

To validate the performance of IncLSTM, ARMA-ONS and BiLSTM were compared. Fig. 5 showed that ARMA-ONS is faster in training on a large dataset, however Table 3 indicated that prediction accuracy of ARMA-ONS is far lower than IncLSTM and Bi-LSTM. Comparing with Bi-LSTM, for small data size, IncLSTM does not have obvious advantage, even take a little longer training time; however, as the data size increases, performance advantage of IncLSTM is increasingly obvious. IncLSTM decreases training time by 18.8% in average. Overall, IncLSTM has better and more stable training performance, especially for large size of data.

5.3. Transfer learning optimization analysis

The training time of a single weak learner is shown in Table 2. As shown in Table 2, the training time for a single epoch for weak learner 2 is the same as that for weak learners 3, 4, and 5, while slightly higher than that for weak learner 1. VeryFastKMM-Tradaboost based transfer learning method was used in IncLSTM to train weak learners. After the weak learner 2, the data of the two datasets were combined for training, and multiple iteration cycles were carried out. Since FL-Share algorithm and early stopping strategies are



	Group1	Group2	Group3	Group4	Group5
ARMA-ONS	196	290	397	498	583
Original BiLSTM	1654	3346	5360	7326	11086
OnlineLSTM	1672	3834	5230	6276	7782

Fig. 5. Comparisons of training performance.

Table 2
The training time of IncLSTM.

Dataset	Stop Epoch Rounds	Single Epoch time (s)	Update time of weak learner (s)	Update time of strong learner (s)	Time interval (s)
Data1	50	32	1640	1672	32
Data2	100/50	42	2124	2162	38
Data3	100/31	44	1352	1396	44
Data4	100/23	44	1002	1046	44
Data5	100/33	44	1462	1506	44

Table 3
Prediction accuracy of different methods.

$Group_1$	EV	MAE	MSE	RMSE	R^2
ARMA-ONS	0.685	29.133	2171.324	46.597	0.651
Original BiLSTM	0.919	11.896	525.095	22.915	0.919
Standard-BiLSTM	0.901	11.743	528.529	22.99	0.9
Instance-based-Transfer-BiLSTM	0.912	11.019	494.049	22.227	0.912
IncLSTM	**0.928**	**10.073**	**454.782**	**21.326**	**0.928**
$Group_2$	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
ARMA-ONS	0.58	32.65	2750.693	52.447	0.57
Original BiLSTM	0.918	11.704	547.483	23.398	0.918
Standard-BiLSTM	0.891	12.688	582.101	24.127	0.89
Instance-based-Transfer-BiLSTM	0.909	11.775	543.295	23.309	0.909
IncLSTM	**0.922**	**10.622**	**501.442**	**22.393**	**0.922**
$Group_3$	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
ARMA-ONS	0.445	54.387	6231.745	78.941	0.423
Original BiLSTM	**0.919**	13.465	564.595	23.761	0.91
Standard-BiLSTM	0.895	11.693	554.594	23.55	0.895
Instance-based-Transfer-BiLSTM	0.904	11.099	557.118	23.603	0.904
IncLSTM	0.913	**10.176**	**531.845**	**23.062**	**0.913**
$Group_4$	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
ARMA-ONS	0.536	34.636	3079.34	55.492	0.52
Original BiLSTM	**0.916**	13.524	613.47	24.768	0.91
Standard-BiLSTM	0.893	12.536	591.606	24.323	0.892
Instance-based-Transfer-BiLSTM	0.904	11.917	579.354	24.07	0.904
IncLSTM	**0.915**	**10.984**	**539.872**	**23.235**	**0.915**
$Group_5$	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
ARMA-ONS	0.58	34.086	2882.146	53.686	0.558
Original BiLSTM	0.919	12.118	539.094	23.206	0.918
Standard-BiLSTM	0.908	12.812	541.181	23.263	0.908
Instance-based-Transfer-BiLSTM	0.908	12.218	538.893	23.214	0.908
IncLSTM	**0.929**	**10.851**	**489.742**	**22.13**	**0.929**

Note: $Group_i = \bigcup_{i=1}^{m} Data_i$

Table 4
Prediction accuracy comparisons on more distant subsets.

$Group_1$→Data6	EV	MAE	MSE	RMSE	R^2
Original BiLSTM	0.921	12.973	555.39	23.567	0.921
IncLSTM	0.93	10.853	497.882	22.313	0.929
$Group_2$→Data6	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
Original BiLSTM	0.921	12.917	563.577	23.74	0.92
IncLSTM	0.929	10.981	500.8	22.379	0.928
$Group_3$→Data6	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
Original BiLSTM	0.929	15.373	601.306	24.522	0.915
IncLSTM	0.93	10.907	494.019	22.227	0.929
$Group_4$→Data6	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
Original BiLSTM	0.923	14.023	577.14	24.024	0.918
IncLSTM	0.929	10.878	490.386	22.145	0.929
$Group_5$→Data6	**EV**	**MAE**	**MSE**	**RMSE**	**R^2**
Original BiLSTM	0.919	12.118	539.094	23.206	0.918
IncLSTM	0.929	10.851	489.742	22.13	0.929

Note: $Group_i = \bigcup_{i=1}^{m} Data_i$

Algorithm 1
Strong Learner Generation Algorithm

Input: Trained weak learners {$f_1(x)$, $f_1(x)$, …, $f_n(x)$}, new data set D_{n+1}
Output: Strong learner $SL_n(x)$

1. Initialize weights of sample i in D_{n+1}, $w_{n+1}(i) = \frac{1}{m_{n+1}}, \forall i, i = 1, 2, \ldots, m_{n+1}$, where m_{n+1} is sample size of D_{n+1}
2. For each weak learners$f_k(x)$: $X \rightarrow Y$, k=1, 2,…, n, take dataset D_{n+1} as input to predict using $f_k(x)$:
3. Calculate maximal error rate on D_{n+1}: $E_k = \max|y_i - f_i(x_i)|$, i=1, 2, 3,…, m_{n+1}
4. Calculate relative error of weak learner $f_k(x)$ over sample i: $e_k(i) = 1 - \exp\left(\frac{-|y_i - f_k(x_i)|}{E_k}\right)$
5. Calculate regression error rate:$e_k = \sum_{i=1}^{m_{n+1}} w_{n+1}(i) * e_k(i)$
6. Calculate standardized error rate:$\beta_k = \frac{e_k}{1 - e_k}$
7. Calculate weak learner rate:$a_k = \frac{1}{2} * \log\left(\frac{1}{\beta_k}\right)$
8. Calculate final rate of each weak learner:$a_k = a_k / \sum_{i=1}^{k} a_k$
9. Generate strong learner:$SL_n(x) = a_1 * f_1(x) + a_2 * f_2(x) + \ldots + a_n * f_n(x)$

Algorithm 2
Sample-based Transfer Learning Algorithm for Weak Learners

Input: historical data set T_{old}, target data set T_{new}, testing data set S, trained weak learner for T_{old} *Learnerold*, and number of iterations N
Output: new regressor f_t
1. Initialize weights:
2. β = *VeryFastKMM*(X_{old},X_{new});
3. Combine training data $T = T_{old} \cup T_{new}$
4. Initialized weight vector:$w = (w_1, w_2, \ldots, w_{n+m})$, where $w = \begin{cases} \beta, i = 1, 2, \ldots, n \\ \max(\beta), i = n+1, n+2, \ldots, m \end{cases}$
5. Set $\rho = \frac{1}{1 + \sqrt{2 * In\frac{n}{N}}}$
6. For each t = 1, 2, …, N
7. Set $p^t = \frac{w^t}{\sum_{i=1}^{n+m} w_i^t}$
8. Taking T and p^t as input, re-train $Learner_{old}$ and get regressor f_t: $X \rightarrow Y$;
9. Calculate error rate e_i of $f_t$over T_{target}, and $e_i = 1 - \exp\frac{-|y_i - f_t(x_i)|}{E_k}$,
where $E_k = \max_{i=1}^{n_{target}} |y_i - ft(xi)|$ then overall regression error rate of *Ttarget* is: $\varepsilon t = \sum_{i=1}^{n_{target}} w_i^t * e_i$;
10. Set $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$
11 Update all weight vector of training data $w_i^{t+1} = \begin{cases} w_i^t * \rho^{|y_i - f_t(x_i)|}, i = 1, 2, \ldots, n \\ w_i^t * \beta_t^{-|y_i - f_t(x_i)|}, i = n+1, \ldots, n+m \end{cases}$
12. Output regressor f_t

Algorithm 3
FL-Sharing transfer learning algorithm for weak learner

Input: dataset D_t, *and* D_{t-1}, Weak learner $Learner_{t-1}$ over data set D_{t-1}
Output: Weak learner $Learner_t$
1. Load weak learner $Learner_1$ over D_1
2. Fix first few layers of $Learner_1$, re-train last layer bi-lstm and neural network layers
3. Link bi-lstm layer of original neural network with new bi-lstm layer through shared layer
4. Define network layers involved in the training
5. Load current data set D_t, and conduct Finetune training for new network
6. Output new learner $Learner_t$

adopted, the training time is bounded in a stable range. More importantly, multiple weak learners continuously update strong learners, and consequently improve training efficiency effectively. As shown in Table 2, training strong learner takes much less time than a single weak learner.

5.4. Model prediction accuracy

Table 3 compares the prediction accuracy between the proposed IncLSTM and BiLSTM. As shown in Table 3, IncLSTM improves the BiLSTM by 0.32% and 0.7% in terms of explained_variance and determination coefficient R^2, respectively. The fitting effect of IncLSTM is 0.9214, indicating a strong explanatory ability to the target variable. Also, IncLSTM significantly decreased prediction errors: MAE, MSE, and RMSE, by 15.6%, 9.8%, and 5%, respectively. The results show that IncLSTM predicts the data more accurately.

Algorithm 4
Buffer Algorithm for Weak Learners

Input: maximal size of buffer ρ, tolerance rate γ.
Output:set of weak learners
1. If $i = 1,2,\ldots, \rho$, put new weak learner into buffer
2. If $i=\rho+1,\ldots$
3. For new generated data set D, conduct prediction over D using each weak learner in buffer, and get loss set $e_{list} = \{e_1,e_2,\ldots, e_p\}$
4. Conduct prediction over data set D using new generated weak learner, and get loss e_i
5. Compare e_i with maximal value in loss set e:
6. If $e_i - \max(e_{list}) \leq \gamma$
   remove weak learner with maximal loss value and put current weak learner into buffer
7. Else
8. ignore current weak learner
9. Output set of weak learners

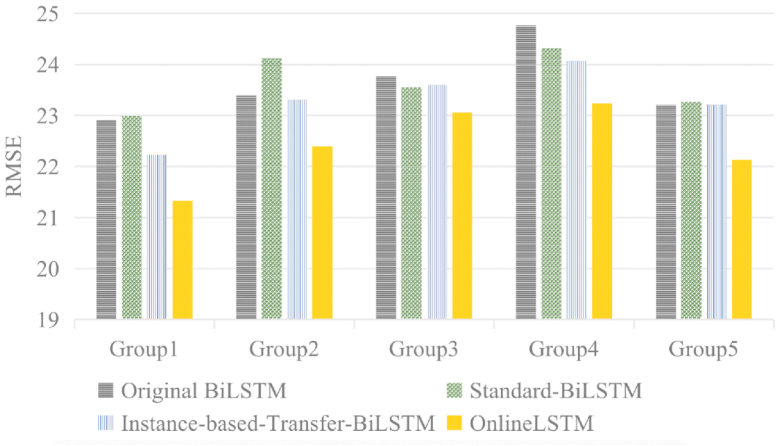


Fig. 6. RMSE loss variation of different weak learner models.

The lastly divided subset is used as a test set to evaluate the robustness of IncLSTM to the increase in data volume. The results are also given in Table 4. Table 4 shows that BiLSTM and IncLSTM maintain the prediction accuracy on far-away datasets. However, IncLSTM outperforms BiLSTM in terms of all evaluation indices. In view of the changing trend of model error evaluation indices MAE, MSE and RMSE, IncLSTM can maintain the stability of model prediction better than the original BiLSTM model. Further, the prediction error is decreased as the data volume increase, which indicates that the proposed model can adjust the model parameters by rapidly adapting the changes in data characteristics of the dataset.

5.5. *Evaluation of the weak learner model*

In order to verify the improvement of the model fitting effect of the weak learner module, comparative experiments were conducted on BiLSTM, Standard-BiLSTM, Instance-based-Transfer-BiLSTM, and IncLSTM. The experimental results are shown in Table 3.

Table 3 and Fig. 6 show that IncLSTM achieves a fitting degree value of 0.9214 on average and a significantly lower RMSE on a time series dataset at multiple stages. The experimental results draw the followings:

1) Standard-BiLSTM is simply used to train multiple truncated molecular datasets. Even with the fine-tuning mechanism, the gradually changed feature distribution is ignored. Therefore, the prediction error of the standard-BiLSTM is relatively higher than that of the original-BiLSTM. Also, standard-BiLSTM has poor generalization ability.
2) Compared with the Standard-BiLSTM method, the Instance-based-Transfer-BiLSTM method gives lower prediction error, which indicates that adding a historical dataset for sample transfer learning improves the model performance. The prediction error of Instance-based-Transfer-BiLSTM is the same as that of the original offline BiLSTM model.
3) In the proposed Incremental-Integrated-BiLSTM method, the shared layer mechanism is added based on

Instance-based-Transfer-BiLSTM. The proposed model significantly improves the prediction accuracy compared to standard-BiLSTM. Also, IncLSTM shows better performance than the offline original BiLSTM model in terms of the degree of fitting and prediction error thanks to the shared layer mechanism. The shared layer mechanism extracts useful information from the original neural network and integrates it with the current new input information so that the model obtains a strong generalization ability.

In summary, ARMA-ONS, which is the traditional online time series prediction model, trains the model extremely fast on a large dataset. However, ARMA-ONS gives poor prediction accuracy due to the limitation of its model algorithm. The traditional BiLSTM model can better model the time-series data of the big dataset, but its offline training requires a long training time. Compared with the traditional offline BiLSTM neural network, the proposed model outperforms in terms of the prediction accuracy and training efficiency through the incremental prediction, which shows that IncLSTM is one efficient and promising solution for large size of training data.

6. Conclusion

Traditional LSTM model is difficult to adapt to the model evolution update of massive time series data. To address the limitation, this paper proposed the ensemble LSTM neural network model with incremental learning. Our proposal implements the dynamic updating and evolution of the model. In this proposed method, the time series data is sliced into several sub-datasets and trained in parallel to form multiple LSTM weak learners. The sequential weak learners are combined to construct a strong learner. In the construction of weak learners, sample transfer learning based on the Tradaboost algorithm and FL-Share transfer learning are used. In the strong learner integration process, the weights of several weak learners are adjusted according to their predicted losses. Also, the buffer pool strategy is used to avoid the infinite growth of the number of weak learners, maintaining the stability of the model. Experiments prove the prediction accuracy and training efficiency of the proposed incremental learning algorithm.

CRediT authorship contribution statement

Huiju Wang: Conceptualization, Methodology, Validation, Writing – review & editing. **Mengxuan Li:** Software, Data curtion, Writing – original draft. **Xiao Yue:** Supervision, Writing – review & editing, Funding acquisition, Project administration.

Declaration of Competing Interest

None.

References

[1] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
[2] Smagulova K, James AP. A survey on LSTM memristive neural network architectures and applications. Eur Phys J Spec Top 2019:2313–24.
[3] Kemker R, McClure M, Abitino A, Hayes TL, Kanan C. Measuring catastrophic forgetting in neural networks. AAAI 2018:3390–8.
[4] Anava O, Hazan E, Mannor S. Online learning for time series prediction. J Mach Learn Res 2013;30:172–84.
[5] Polikar R, Upda L, Upda SS, Honavar V. Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cybern Part C 2001; 31(4):497–508.
[6] Shen Y, Zhu Y, Song X. Fast learn++. NSE algorithm based on sliding window. Pattern Recognit Artif Intell 2018;32(5):1737–45.
[7] Gangardiwala A, Polikar R. Dynamically weighted majority voting for incremental learning and comparison of three boosting based approaches. In: Proceedings of IEEE international joint conference on neural networks; 2005. p. 1131–6.
[8] Shan G, Xu s, Yang Li, Jia S, Xiang Y. Learn#: a Novel incremental learning method for text classification. Expert Syst Appl 2020;147:113198.
[9] Xiang Y, Miao Y, Chen J, Xuan Q. Efficient incremental learning using dynamic correction vector. IEEE Access 2020;8:23090–9.
[10] Hazan T, Sabach S, Voldman S. Stochastic proximal linear method for structured non-convex problems. Optim Methods Softw 2020. https://doi.org/10.1080/10556788.2020.1754413.
[11] Liao S, Zhang X. Online kernel selection via tensor sketching. CIKM; 2019. p. 801–10.
[12] Sahoo D, Pham Q, Lu J, Hoi SCH. Online deep learning: learning deep neural networks on the fly. IJCAI; 2018. p. 2660–6.
[13] Sarwar S, Ankit A, Roy AK. Incremental learning in deep convolutional neural networks using partial network sharing. IEEE Access 2020;8:4615–28.
[14] Mosca A, Magoulas GD. Deep incremental boosting. CoRR 2017. abs/1708.03704.
[15] Katuwal R., Suganthan P. N. and Tanveer M., "Random vector functional link neural network based ensemble deep learning", arXiv, 2019, preprint arXiv: 1907.00350.
[16] Li Y, Wang Y, Liu Q, et al. Incremental semi-supervised learning on streaming data. Pattern Recognit 2019;88:383–96.
[17] Chang J, Cao F, Zhou A. Clustering evolving data streams over sliding windows. J Softw 2007;(04):137–50.
[18] Aggarwal C C, Philip S Y, Han J, et al. A framework for clustering evolving data streams. VLDB 2003:81–92.
[19] Lu H, Li Y, Mu S, Wang D, Kim H, Serikawa S. Motor anomaly detection for unmanned aerial vehicles using reinforcement learning. IEEE Internet Things J 2018; 5(4):2315–22.
[20] Leo B. Bagging predictors. Mach Learn 1996;24(2):123–40.
[21] Ghaeini R, Hasan S A, Datla V, et al. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. NAACL-HLT 2018:1460–9.
[22] Xu G, Meng Y, Qiu X, et al. Sentiment analysis of comment texts based on BiLSTM. IEEE Access 2019;7:51522–32.
[23] Pan S J, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng 2009;22(10):1345–59.
[24] Dai W, Yang Q, Xue GR, et al. Boosting for transfer learning. In: Proceedings of the 24th international conference on Machine learning. ACM; 2007. p. 193–200.
[25] Chandra S, Haque A, Khan L, et al. Efficient sampling-based kernel mean matching. ICDM 2016:811–6.
[26] Li H. Research and application of case-based transfer learning technology. Phd Thesis. Wuhan University; 2018.
[27] Rosenfeld A, Tsotsos JK. Incremental learning through deep adaptation. IEEE Trans Pattern Anal Mach Intell 2020;42(3):651–63.
[28] UCI Dataset. https://archive.ics.uci.edu/ml/index.php, [2019-09-20].