A Deep Learning Framework for Univariate Time Series Prediction Using Convolutional LSTM Stacked Autoencoders

Aniekan Essien
Zienkiewicz Centre for Computational Engineering (ZCCE)
College of Engineering, Swansea University
Swansea, United Kingdom
a.e.essien@swansea.ac.uk

Cinzia Giannetti
Zienkiewicz Centre for Computational Engineering (ZCCE)
College of Engineering, Swansea University
Swansea, United Kingdom
c.giannetti @swansea.ac.uk

***Abstract*—This paper proposes a deep learning framework where wavelet transforms (WT), 2-dimensional Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) stacked autoencoders (SAE) are combined towards single-step time series prediction. Within the framework, the input dataset is denoised using wavelet decomposition, before learning in an unsupervised manner using SAEs comprising bidirectional Convolutional LSTM (ConvLSTM) layers to predict a single-step ahead value. To evaluate our proposed framework, we compared its performance to two (2) state-of-the-art deep learning predictive models using three open-source univariate time series datasets. The experimental results support the value of the approach when applied to univariate time series prediction.**

Keywords—time series prediction, stacked autoencoders, data science, deep learning, Convolutional-LSTM

I. INTRODUCTION

Time series prediction is typically considered as a challenging task, which applies to many fields of endeavour. In the financial sector, analysts are tasked to predict stock market prices, exchange rates, loan default rate, or stock market indices. For instance, electricity generation and distribution companies require predictions of load demand to enable efficient electricity generation and distribution. Similarly, in traffic management, there is a need to predict one or more traffic parameters to manage the traffic situation effectively. A common theme in the problems listed above revolves around the need to analyse past and/or current observations of a variable (or variables) to predict the values of future observations. Due to the broad significance and multidisciplinary application of time series, there has been an increase in research studies targeted at the development of techniques for making accurate time series predictions.

A number of techniques and algorithms have been proposed for time series prediction, such as linear models, which include (but are not limited to) Auto-Regressive Integrated Moving Average (ARIMA) and its variants [1], support vector machines [2], statistical analysis [3] and, more recently, deep non-linear neural network algorithms like Recurrent Neural Networks (RNN) [4], LSTMs [5] and CNNs [6], which have been applied in many areas such as in financial prediction [7], [8], traffic prediction [9]–[11], machine fault prognosis/diagnosis [12] and anomaly detection [13], [14]. Although ARIMA and ARIMA-based model variants such as Seasonal ARIMA (SARIMA) [1], Vector ARIMA (ARIMAX) [15] have shown promising signs when applied towards univariate and multivariate time series prediction, they however show vulnerabilities when applied to non-linear, sequential, or time series data, such as traffic and stock prediction [9]. In recent literature, the trend is inclined towards the use of deep learning algorithms for short to medium-term time series forecasting. This is mainly due to the performance of such models in extracting robust features in time series data while preserving the temporal/sequential dimension of time series data, which results in improved predictive accuracy with compared to traditional and non-linear machine learning algorithms [16], [17].

Broadly speaking, three main approaches for deep learning are popular in literature – convolutional neural networks (CNNs) [18], deep belief networks (DBNs) [19], and stacked autoencoders (SAEs) [20]. Extant studies have exhaustively studied these three broad categories, with a number of novel algorithms, frameworks, and approaches proposed in the last decade. For instance, a stacked autoencoder framework was proposed in [21] for stock market prediction. Similarly, the authors in [22] presented a bidirectional convolutional LSTM model for hyperspectral image classification. While recent research has resulted in many accurate time series prediction algorithms, there are still opportunities for the development and application of new algorithms and methodologies. This paper, therefore, aims to contribute to the growing time series prediction research community by proposing a deep learning framework comprising bidirectional ConvLSTM stacked autoencoders for univariate time series prediction.

Our proposed framework comprises of three distinct stages: (i) Wavelet Decomposition/Transformation, (ii) Stacked AutoEncoders (SAE), and (iii) 2-Dimensional bidirectional ConvLSTMs. The SAE architecture, which also serves as a dimensionality reduction technique, performs feature extraction and sequence learning from the time series input data in an unsupervised manner. The approach proposed in this study differs from extant related studies by utilizing a

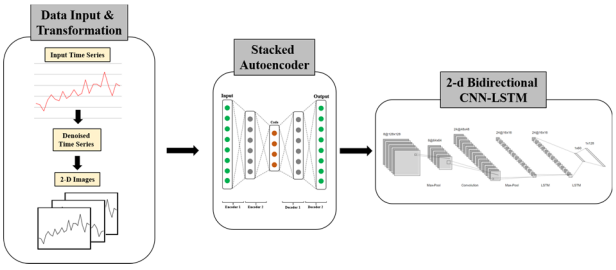

Fig. 1: Flowchart of proposed framework

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) project EP/S001387/1 and the European Regional Development Funds projects IMPACT and Supercomputing Wales.

978-1-7281-1862-8/19/$31.00 ©2019 IEEE

2-dimensional bidirectional ConvLSTM architecture in which the input time series are learned in an unsupervised manner by the 2-dimensional Convolutional stacked autoencoder layers. The hybrid ConvLSTM architecture is a well-known technique [23] for accurately predicting time series. LSTMs are a type of recurrent neural networks (RNN) that have forget gates, which provide control to the model in terms of what to forget/remember in a time series or sequential learning process [24]. Studies [10], [13] have shown that LSTMs outperform traditional RNNs in time series prediction, which justifies its use within our framework. Wavelet decomposition can also be used as a denoising strategy in time series data and has been widely used in filtering/mining signals [25]. Within our proposed framework, we use WT for denoising the input time series before learning using the ConvLSTM-based deep learning framework. The proposed model, which is a combination of WT, SAE, and 2D-ConvLSTM, will be hereafter referred to as the *2DWSAE*. The novelty of our approach lies in the application of 2-dimensional bidirectional ConvLSTM bidirectional stacked autoencoders and wavelet denoising towards time series prediction.

To test our framework, we selected three (3) openly available univariate time series datasets: (i) The Chickenpox dataset, (ii) minimum daily temperatures dataset and (iii) PeMS daily traffic flow dataset. The dataset contains 51,840 observations of traffic flow. It is important to mention here that our framework is tested for generalization on the three datasets, as opposed to extant studies that validate on a single dataset. We also benchmark the performance of *2DWSAE* against two (2) state-of-the-art time series prediction models in literature. The models are: (i) CNN-LSTM Autoencoder (CNN-LSTM-SAE) and (ii) 2D Convolutional LSTM (2DConvLSTM). The CNN-LSTM-SAE is a gold-standard time series prediction model that has been applied in various field such as in machine health [26], solar irradiance forecasting [27], stock prediction [28] and fault diagnosis [26], [29]. Similarly, the ConvLSTM has been applied in time series prediction [30]–[32].

The remainder of this paper is organized as follows. Section II presents our framework methodology and technical background of key concepts. In section III, we describe the datasets used, while Section IV discusses the experimental setup including model description and implementation environment. The results are presented in Section V, while we conclude and discuss future work in Section VI.

II. METHODOLOGY

Fig. 1 depicts the flowchart of the proposed *2DWSAE* model for single-step prediction from time series input data. The framework comprises three stages: (1) input data pre-processing and transformation using wavelet decomposition, which is a denoising strategy applied to decompose the time series and eliminate noise; (2) stacked autoencoders, which represent a deep learning architecture by training on a dataset in an unsupervised learning method; (3) a bidirectional 2D ConvLSTM in order to extract features of the input data and learn sequentially. We present further descriptions about the distinct stages in the next paragraphs.

A. Continuous Wavelet Transformation (CWT)

A wavelet can be defined as a function with zero mean and one that is localized in both the frequency and time domains [33]. Wavelet transformation is typically applied in time series for data denoising as it is able to handle non-stationary time series data [33]. A key characteristic of wavelet transformation is the simultaneous analysis of frequency components with time, a property lacking in the also popular Fourier transformation [34]. The Fourier transform uses a series of sine-waves with different frequencies in order to analyze an input signal. In other words, the input signal is represented via a linear combination of sine-waves. The wavelet transform uses a series of 'wavelets', which are small waves in order to decompose or transform the dataset. In this way, the wavelet transform is able to preserve the time factor of the time series data, instead of decomposing the input data into a frequency-based output as is applicable in the Fourier transform. Mathematically, the WT of a continuous time series $f(t)$ is given by the following equation:

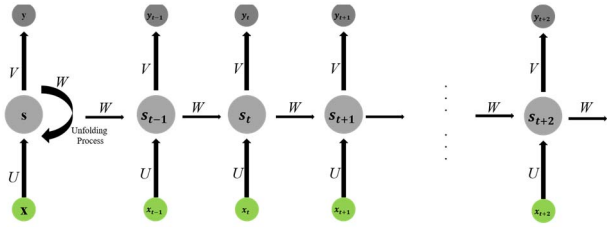

Fig. 2: Structure of the RNN

$$X_\omega(a,b) = \frac{1}{\sqrt{a}} \emptyset\left(\frac{t-b}{a}\right) \quad (1)$$

Where $\emptyset(t)$ represents a continuous function both in the time and frequency domain, a and b represent the scale and translation factor respectively.

B. Deep Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) [4] constitute a class of traditional neural networks that extend the functionality of neural networks by preserving the temporal dimension of sequential data (for instance, time series). Unlike traditional feedforward neural networks, RNNs add a loop or 'recurrent' component in order to connect the neuron to itself and unfold it as many times such that it is able to produce a probability distribution in the sequential data. RNNs have hidden states that are updated by the sequential information obtained from a time series with an output that is dependent on the hidden states. Fig. 2 shows the mechanism of an RNN being unfolded into a network. U and V represent the weights of the hidden layer and output layer respectively, while W represents the transition weights of the hidden state.

In Fig. 2, let us consider an input vector x_t at time t. Also, let s_t represent the hidden state of the RNN at time t, which is calculated as the element-wise product of the input vector and the previous hidden state. Therefore, the hidden state at time t, given its previous hidden state h_{t-1} is calculated using (2) below.

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b) \quad (2)$$

Where W_{hx} represents the weight between the input and recurrent hidden nodes, W_{hh} represents the weight between the recurrent node and the previous time step of the hidden node itself, b and σ represent bias and non-linear (sigmoid) activation respectively.

Although RNNs perform better in sequential datasets, they still have issues yet to be addressed [35]. As can be seen from equation (2), the recurrent hidden node with respect to itself

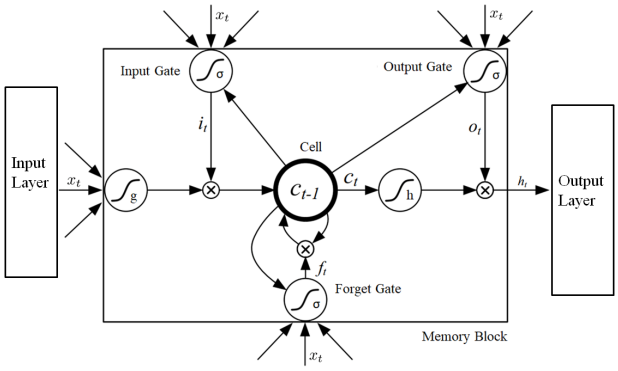

Fig. 3: Single memory cell LSTM neural network

approaches zero as the time interval increases. This leads to a situation referred to as *diminishing gradient* [36], [37], a vulnerability that is encountered when using RNNs for long-term time series modelling. To resolve this, German engineers Hochereiter and Schmidhuber [24] proposed the Long Short-Term Memory (LSTM) RNN, which had the primary objective of modelling long-term time dependencies in time series. The LSTM model replaced the recurrent hidden unit with a memory cell. The memory cell contains a node with itself connected to the recurrent edge of a fixed weight node, thereby ensuring that the gradient survives longer time steps without vanishing. The representation of the LSTM having one memory block is depicted in Fig. 3. It can be observed from Fig. 3 that the memory block contains input, output, and forget gates, which respectively represent write, read, and reset functions on each cell. The multiplicative gates allow the model to store information over long periods, thereby eliminating the vanishing gradient problem commonly observed in traditional neural network models [36].

Consider a time-series input sequence denoted by $x = x_1 + x_2 + x_3 \dots x_t$ and output sequence of $y = y_1 + y_2 + y_3 \dots y_t$, where t is the prediction horizon. The LSTM automatically computes the predicted output in the next time step using the historical information supplied, without predetermining the lag observations to use. The following set of equations are performed by the model and enables the model to predict the output variable:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (4)$$

$$c_t = f_t c_{t-1} + i_t g(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t h(c_t) \quad (7)$$

Where W and b represent the weight matrix and bias vector respectively and σ(.) denotes a standard logistic sigmoid function defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (8)$$

$$g(x) = \frac{4}{1+e^{-x}} - 2 \quad (9)$$

$$h(x) = \frac{2}{1+e^{-x}} - 1 \quad (10)$$

Where $g(.)$ and $h(.)$ are the respective transformations of the sigmoid function above. The variables $i, f, o,$and c are the input gate, forget gate, output gate, and cell activation vector respectively. This particular characteristic makes LSTM a reliable and accurate model for use in time series analysis and traffic prediction.

C. Convolutional LSTM

A convolutional LSTM or ConvLSTM is a variant of the LSTM that replaces the fully-connected layer operators with convolutional operators [38]. Fig. 4 represents the structure of the ConvLSTM model. The left hand image represents the conventional LSTM memory cell, which appears to be zoomed in on the convolution layers of the right hand side image. Within the LSTM memory cell, the $\oplus$ and $\otimes$ operators refer to the matrix addition and dot product operators respectively. Therefore, by replacing the convolution operators with an LSTM memory cell, the ConvLSTM is able to know what information is to be 'remembered' or 'forgotten' from the previous cell state, using its forget gate. Similarly, the ConvLSTM also decides what information is to be stored in the present cell state. The process of the ConvLSTM is described in a similar manner to the equations (3) - (7) used for the LSTM memory cell computation.

D. Stacked Autoencoders

In its simplest form, an autoencoder is a type of feedforward network in which the input is the same as the output. In other words, autoencoders compress the input vector into a lower-dimensional code and attempt to reconstruct the output from this given representation. Fig. 5 represents the basic structure of an autoencoder. As can be seen, the autoencoder consists of three major components: the encoder, the code, and the decoder. The functions of these elements are as easy as their respective names suggest. The encoder compresses the input to a 'latent space' to produce the code, which is then decoded by the decoder. In other words, the output is reconstructed from the code using the decoder.

The stacked autoencoder is a stack of autoencoders and, just like autoencoders, learn in an unsupervised manner. In this study, the stacked autoencoders are represented by the bidirectional 2-dimensional ConvLSTM architectures discussed in the previous section and depicted in fig. 4. The learning process involves layer-wise training in order to minimize the error between input and output vectors. The activation function applied within the hidden units is the Rectified Linear Unit (ReLU), which is mathematically defined by (11).

$$g(z) = \max\{0, z\} \quad (11)$$

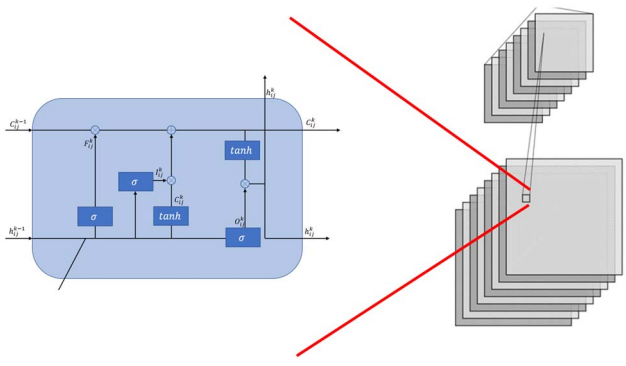

Fig. 4: Basic structure of the Convolutional LSTM

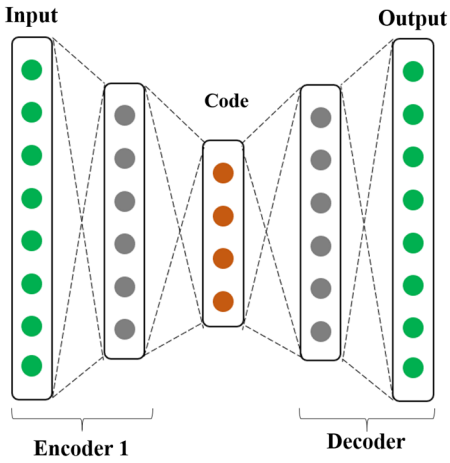

Fig. 5: Basic structure of the autoencoder

The subsequent layer of the autoencoder is the hidden layer of the previous one, with each of the layers using an optimization function, which is the squared reconstruction error J of the individual autoencoder layer described in (12).

$$\begin{aligned} &\operatorname{argmin}_{W_1,b_1,W_2,b_2}[J] \\ &= \operatorname*{argmin}_{W_1,b_1,W_2,b_2}\left[\frac{\sum_{i=1}^{m}\|x_i - x_i'\| + J_{wd} + J_{sp}}{2}\right] \end{aligned} \tag{12}$$

Where J represents the squared reconstruction error of the single autoencoder layer, x_i and x_i' respectively represent the ith value of the input vector as well as its corresponding reconstructed version. m represents the training dataset size, corresponding to the length of the input time series.

III. Data Description

In this section, we present details about the dataset description, as well as the data resources.

A. Chickenpox dataset

The chickenpox dataset is a univariate timeseries of monthly chickenpox instances over a period of 42 years. The chickenpox dataset is obtainable online[1].

B. Daily minimum temperature in Melbourne Australia

This dataset contains 3,650 observations of the daily minimum temperature in Melbourne, Australia from 1981-1990. This dataset can be obtained online[2].

C. PeMS daily traffic flow dataset

This dataset contains 5-min aggregated traffic flow from the California Department of Transportation PEMS website. The dataset contained 51,840 observations of traffic flow. This dataset is publicly available online[3]. The sample period spanned from 01/01/2013 to 30/06/2013, representing 180 days of 5-min aggregated traffic flow.

IV. Experimental Setup

For this study, we used the prediction setup described in [9]. In particular, the procedure involves the application of an overlapping sliding window approach for reconstructing the input time series from a univariate time series to a supervised learning format. A sliding window size of 12 was used, with the last 12 lags (previous time steps) used as input timesteps in order to predict a single step ahead. The time shift factor was also unity, with the sliding window being shifted a singl step forward per window/observation. The subsequent dataset was then split into training and testing datasets using a ratio of 70:30 for train and test partitions respectively. Finally, the respective datasets were exposed to the models for learning to perform single-step prediction on the test subsets of the respective datasets.

A. Model Description

The proposed model incorporates a bidirectional 2DConvLSTM stacked autoencoder architecture. The LSTMs used within the framework are bi-directional (the hidden layer of the previous layer serving as the visible/input layer of the next layer) each had 200 hidden units. For all inter-connected layers (except the output layer), the activation function utilized was the Rectified Linear Unit (ReLU), which introduced non-linearity to the learning process, and is defined according to equation (11). The performance of deep learning networks is dependent on key parameters that must be predetermined. Some of the parameters are: model depth (i.e. hidden layer size), model width (i.e. number of units within each hidden layer), number of epochs/iterations for training, batch size, dropout rate, and optimizer. The process of selecting the combination of these parameters is not standardized and so is more of an art than a science, with many data scientists opting towards a 'brute force' approach. In order to identify the optimal set of hyper-parameters to apply, we developed a grid search framework. This enabled a quicker and more efficient method for arriving at the optimum set of parameters.

In order to evaluate our proposed framework, we benchmarked the performance of our model against state-of-the-art predictive models (see Section 2). Each of the models were trained to learn the features provided within the time series datasets. The overall prediction algorithm is presented in Algorithm 1.

Algorithm 1: Algorithm for 2-dWSAE

Input: observed sequences $X = x_1, x_2, x_3, \dots, x_T$
Output: single-step predicted value Y= y_{t+1}

1. Split Xinto training and testing data with 70:30 ratio
2. Perform wavelet transformation using Haar distance
3. Randomly initialize model parameters weight w_t and bias c
4. **for** each iteration **do**
5. Select lookback steps/batch size b
6. Generate lookback sequence as $x_1, x_2, \dots, x_b$
7. feed sequential batch into the 2-D CNN-LSTM SAE
8. Train using forward greedy layer-wise and bi-directional processing
9. Obtain actual value and compute error (i.e. MSE)
10. Update model learning by backpropagation algorithm using optimizer (Adam in this case), minimizing loss function $L = \frac{1}{n}\sum_{i=1}^{n}|y^i - \hat{y}^i|$, where $\hat{y}^i$ is the predicted sequence in the i^{th}layer.
11. End **for**
12. Reiterate until training set is exhausted
13. Return prediction output sequence Y.

B. Model Performance Evaluation

In terms of model evaluation, we adopted a technique referred to as walk-forward validation or back testing [39].

[1]Dataset available at: https://catalog.data.gov/dataset?tags=chickenpox

[2] Dataset available at: https://datamarket.com/data/

[3] Dataset available at: http://pems.dot.ca.gov

Traditional prediction evaluation methods such as *k*-fold cross validation or train-test splitting do not work directly when applied to time series data, due to the fact that these evaluation methods assume there is no relationship between the observations, which is not the case with time series data, where the sequential dimension needs to be preserved.

For model accuracy evaluation, we applied three accuracy evaluation metrics – Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Symmetrical Mean Absolute Percentage Error (sMAPE), which are all defined by the respective equations below.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{13}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i^2)} \tag{14}$$

$$sMAPE = \frac{200}{n}\sum_{i=1}^{n}\left|\frac{x_i - y_i}{x_i + y_i}\right| \tag{15}$$

Where $e_i, i = 1, 2, \ldots n$ represents n samples of modal errors, x_i and y_i respectively represent the input and output values.

C. Implementation Environment

The experiment environment used for this study was on a single machine in Windows 10 Operating System with Intel® Xeon® E-2146G CPU @ 3.50GHz, 32-GB Memory, and NVIDIA Quadro P2000 GPU. The GPU is used for accelerated model training due to large computation demand in deep learning models. The development was performed using Python 3.6.8, R version 3.5.1, and Tensorflow 1.12.0.

V. Results

For each of the three datasets, we aimed at carrying out single-step prediction of the input variable, for instance the daily minimum temperature in the daily minimum temperatures dataset. Fig. 5 illustrates the results of the predictive models on the respective datasets.

As can be seen from the results, our proposed model outperformed the other models in terms of prediction accuracy, with the CNN-LSTM-SAE being the close second. The results show only a marginal improvement in the CP dataset, which can be attributed to the fact that the dataset size is relatively small, having only 498 observations. Deep learning models are known to be data-intensive and mostly leverage on massive datasets and sometimes tend to perform poorly with small to medium datasets [40].

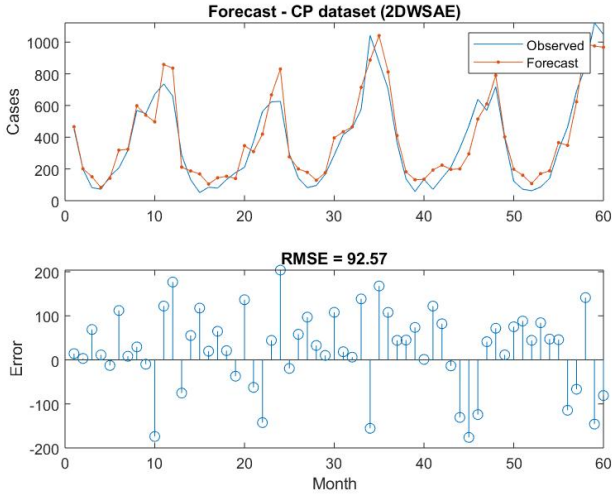

Fig. 6: Predictive performance of 2DWSAE on the chickenpox dataset

TABLE I. Summary of Results

Model ID	Metric	CP	Temp	PeMS
Our Model	MAE	75.24	1.85	10.06
	RMSE	92.57	2.43	13.95
	sMAPE	0.271	0.137	0.187
CNN-LSTM-SAE	MAE	80.09	2.33	10.52
	RMSE	115.42	2.95	13.94
	sMAPE	0.274	0.180	0.190
2DConvLSTM	MAE	96.45	1.96	10.06
	RMSE	140.43	2.530	13.91
	sMAPE	0.415	0.146	0.190

VI. Conclusion

This paper has presented a deep learning framework for single-step time series prediction incorporating wavelet transformation, 2-dimensional convolutional LSTM deep neural networks, and stacked autoencoders. The procedure for the framework actualization is as follows: the input time series is first denoised using wavelet transformation by computing the Haar wavelet. Secondly, the denoised time series is converted into a sequence of 2-dimensional images. This is done to make it suitable for image-recognition CNN networks, which are skilled at image recognition. The proposed framework has been tested using three open source datasets: (1) chickenpox dataset, (2) Daily minimum temperatures in Melbourne Australia, and (3) PeMS traffic flow dataset. In addition, we have compare our framework with state-of-the-art models including 2-dimensional ConvLSTM and CNN-LSTM-SAE models. The results substantiate the value of our proposed framework.

Further research work is identified in applying transfer learning using open-source pre-trained image recognition models such as the VGG16 [41], Squeezenet [42], or AlexNet [43]. Furthermore, the model could be evaluated on additional time series datasets.

References

[1] S. C. Hillmer and G. C. Tiao, "An ARIMA-Model-Based Approach to Seasonal Adjustment," *J. Am. Stat. Assoc.*, vol. 77, no. 377, pp. 63–70, Mar. 1982.

[2] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel time prediction with support vector regression," in *IEEE Conference on Intelligent Transportation Systems*, 2004, pp. 276–281.

[3] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio, "The impact of rainfall and temperature on peak and off-peak urban traffic," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11030 LNCS, pp. 399–407.

[4] P. Rodriguez, J. Wiles, and J. L. Elman, "A Recurrent Neural Network that Learns to Count," *Conn. Sci.*, vol. 11, no. 1, pp. 5–40, 1999.

[5] S. Hochreiter, J. S.-A. in neural information, and undefined 1997, "LSTM can solve hard long time lag problems," *papers.nips.cc*.

[6] S. Lawrence, C. Giles, A. Tsoi, A. B. neural networks, and undefined 1997, "Face recognition: A convolutional neural-

network approach," *cs.cmu.edu*.

[7] K. Chen, Y. Zhou, F. D.-C. on B. D. (Big Data), and undefined 2015, "A LSTM-based method for stock returns prediction: A case study of China stock market," *ieeexplore.ieee.org*.

[8] E. Chong, C. Han, F. P.-E. S. with Applications, and undefined 2017, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," *Elsevier*.

[9] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio, "Improving Urban Traffic Speed Prediction Using Data Source Fusion and Deep Learning," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2019, pp. 1–8.

[10] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. Part C Emerg. Technol.*, vol. 54, pp. 187–197, 2015.

[11] Y. Jia, J. Wu, M. Ben-Akiva, R. Seshadri, and Y. Du, "Rainfall-integrated traffic speed prediction using deep learning method," *IET Intell. Transp. Syst.*, vol. 11, no. 9, pp. 531–536, 2017.

[12] G. Yue, G. Ping, and L. Lanxin, "An End-to-End model based on CNN-LSTM for Industrial Fault Diagnosis and Prognosis," in *IC-NIDC 2018*, 2018, pp. 274–278.

[13] I. Pankaj Malhotra1, Lovekesh Vig2, Gautam Shroff1, Puneet Agarwal1 1- TCS Research, Delhi and I. 2- Jawaharlal Nehru University, New Delhi, "Long Short Term Memory Networks for Anomaly Detection in Time Series," *Proc. 2015 Netw. Distrib. Syst. Secur. Symp.*, 2015.

[14] B. Lindemann, F. Fesenmayr, N. Jazdi, and M. Weyrich, "ScienceDirect Anomaly Detection in Discrete Manufacturing Using Self-Learning Approaches," vol. 00, no. July, pp. 18–20, 2018.

[15] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transp. Res. Part C Emerg. Technol.*, 2011.

[16] Y. Bengio, A. Courville, P. V.-I. transactions on pattern, and undefined 2013, "Representation learning: A review and new perspectives," *ieeexplore.ieee.org*.

[17] Ian Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. 2017.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural information processing systems*, 2012, pp. 1097–1105.

[19] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[20] Y. Bengio, P. Lamblin, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural information processing systems*, 2007, pp. 153–160.

[21] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory.," *PLoS One*, 2017.

[22] Q. Liu, F. Zhou, R. Hang, X. Y.-R. Sensing, and undefined 2017, "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification," *mdpi.com*.

[23] Y. Guo, Z. Wu, and Y. Ji, "A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction," in *Proceedings - 2017 3rd International Conference on Big Data Computing and Communications, BigCom 2017*, 2017.

[24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] M. Aminghafari, N. Cheze, J. P.-C. S. & Data, and undefined 2006, "Multivariate denoising using wavelets and principal component analysis," *Elsevier*.

[26] R. Zhao, R. Yan, J. Wang, K. M.- Sensors, and undefined 2017, "Learning to monitor machine health with convolutional bi-directional LSTM networks," *mdpi.com*.

[27] F. Wang *et al.*, "Wavelet Decomposition and Convolutional LSTM Networks Based Improved Deep Learning Model for Solar Irradiance Forecasting," *mdpi.com*.

[28] S. Liu, C. Zhang, J. M.-I. C. on N. Information, and undefined 2017, "CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets," *Springer*.

[29] H. Pan, X. He, S. Tang, F. M.-S. V. of, and undefined 2018, "An Improved Bearing Fault Diagnosis Method using One-Dimensional CNN and LSTM.," *search.ebscohost.com*.

[30] Z. Yuan, X. Zhou, and T. Y. SIGKDD, "Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 984–992.

[31] Y. Liu, H. Zheng, X. Feng, and Z Chen, "Short-term traffic flow prediction with Conv-LSTM," in *9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017, pp. 1–6.

[32] S. Kim, S. Hong, M. Joh, and S. Song, "DeepRain: ConvLSTM Network for Precipitation Prediction using Multichannel Radar Data," *ArXiv Prepr.*, vol. 1711., no. 02316, Nov. 2017.

[33] D. Percival and A. Walden, *Wavelet methods for time series analysis*. 2006.

[34] P. Bloomfield, *Fourier analysis of time series: an introduction*. 2004.

[35] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[36] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. IEEE Press, 2001.

[37] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[38] S. Xingjian, Z. Chen, H. Wang, … D. Y.-A. in neural, and undefined 2015, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *papers.nips.cc*.

[39] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[40] P. Poonia, V. K. Jain, and A. Kumar, "Short Term Traffic Flow Prediction Methodologies: A Review," *Mody Univ. Int. J. Comput. Eng. Res.*, vol. 2, no. 1, pp. 37–39, 2018.

[41] K. Simonyan, A. Z. preprint arXiv:1409.1556, and undefined 2014, "Very deep convolutional networks for large-scale image recognition," *arxiv.org*.

[42] H. Qassim, D. Feinzimer, and A. Verma, "Residual Squeeze VGG16," May 2017.

[43] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arxiv.org*, vol. 1602, no. 07360, 2016.