Multi-step LSTM Prediction Model for Visibility Prediction

Yunlong Meng[1,✉] , Fengliang Qi[1], Heng Zuo[1], Bo Chen[2], Xian Yuan[2], Yao Xiao[1,✉]
[1]*Em-Data AI Research*, Shanghai, China
[2]*China Eastern Region Air Traffic Management Bureau (ECATMB)* , Shanghai, China
✉*mengyunlong@em-data.com.cn*, ✉*xiaoyao@em-data.com.cn*

***Abstract*—In this paper, we present a deep learning framework with attention mechanism for visibility prediction. We firstly formulate visibility prediction as a temporal prediction problem. An encoder-decoder architecture based network is proposed to generate a multi-step prediction. To adaptively focus on different parts of the input and output sequence, we incorporate input attention and temporal attention into the network. Experiments verify the feasibility of the proposed model. We produce state-of-the-art prediction accuracy (68.9%) on the runway visual range prediction in our customized data set collected at observation stations of the airport.**

I. Introduction

Airport operations are sensitive to visibility conditions. Low-visibility issues may seriously influence the air traffic and lead to flow capacity reduction, as the spacing between two aircraft should be increased. Commonly, the airport low-visibility procedures may come into force when runway visibility falls below the airport-specific thresholds. In this situation, airports will take some special actions to ensure flight safety, i.e., postpone inbound flight landing or divert to alternative airports, and delay or prevent outbound flight taken off. Reliable visibility prediction plays a very important role on aircraft planning and deployment.

Visibility prediction is a challenging task due to the complexity of the physical process. Existing visibility forecasting approaches can be categorized into two classes: physical modeling and statistical modeling. Physical modeling with numerical weather prediction models has been widely explored in last decades, e.g., ALADIN [1], WRF [2], . However, most of these methods cannot achieve a good prediction within the next six hours, as some important parameters cannot be accurately estimated in such a period. Physical modeling methods are also computationally intensive in the inference process, as a highly complex model has been built for implementation. On the other hand, the statistical modeling methods are usually computationally lightweight, especially in the inference process. Since they are data-driven solution, the model parameters are estimated on the train data set and forecasting is implemented on the new data set. Regression models, support vector machines [3], tree-based methods [4], [5], and artificial neural networks [6], [7] have been used for low-visibility forecasting.

In this paper, we formulate the low-visibility forecasting task as a time-series prediction problem. By introducing the convolutional component into the encoder-decoder architecture with attention mechanism, the proposed network model combines the advantages of LSTNet [8] and DA-RNN [9], [10], and achieves the state-of-the-art prediction accuracy (68.9%) on the runway visual range prediction.

II. Related works

A. Physical modeling

The development of numerical weather models have made significant progress on visibility forecasting [11]. Goswami *et al.* explore and evaluate the potential of a dynamic fog forecasting system with visibility calculated from observed meteorological fields [12] for benchmark forecasts from an atmospheric mesoscale model. Shatunova *et al.* find that the numerical prediction meteorological parameters, like relative humidity, or the rate and phase of precipitation, can be used for visibility forecasting with synoptic approach [13]. Steeneveld *et al.* evaluate the HARMONIE and Weather Research and Forecasting (WRF) mesoscale models for two contrasting warm fog episodes and find that the boundary-layer formulation is critical for forecasting the fog onset, while the choice of the microphysical scheme is a key element for fog dispersal [14]. Tudor and Martina implement forecasting experiments with the numerical weather prediction model ALADIN and find that sophisticated radiation scheme is related to the visibility [1]. Zhou *et al.* shows that the performance of the low visibility/fog forecasts from the current operational 12 kmNAM, 13 km-RUC and 32 km-WRF-NMM models at the National Centers for Environmental Prediction (NCEP) models is still unsatisfactory [15]. Lin *et al.* conduct a series of numerical simulations to understand the formation, evolution, and dissipation of an advection fog event over Shanghai Pudong International Airport (ZSPD) with the Weather Research and Forecasting (WRF) model [2].

B. Statistical modeling

We review the existed statistical modeling methods for visibility prediction task in this section. Dutta and Chaudhuri develop visibility forecasting model for fog prediction by combining the decision tree algorithm and artificial neural network approach together at the airport of metropolis of India [4]. Bartokov *et al.* employ decision-tree induction method to build a mode for fog events nowcasting in the coastal

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

desert area of Dubai [5]. Dietz *et al.* tree-based ensemble statistical models based with highly-resolved meteorological observations for low-visibility procedure states forecasting [16]. Kneringer develops an ordered logistic regression (OLR) model based probabilistic forecasting model of low-visibility procedure states at Vienna International Airport [17]. Ortega *et al.* presents an exploratory study of using machine learning algorithms for visibility conditions classification with the data collected from the weather stations in Florida [18]. Bari introduces the machine-learning regression method into Kilometric NWP Model for visibility prediction [19]. Ryerson *et al.* presents a nonparametric ensemble postprocessing approach for short-range visibility predictions in data-sparse regions [20]. Deep neural network based machine learning techniques is revived, since Krizhevsky *et al.* won the champion on ILSVRC competition with a large margin in 2012 with their deep convolutional neural network, AlexNet [21]. Like many other fields, the field of visibility prediction is also benefited from the fast growing deep learning technologies. Artificial neural network model is developed for ceiling and visibility forecasting in [6], [7]. Zhu *et al.* directly apply the deep learning for visibility forecasting in the airport [22]. Palvanov and Palvanov develop "VisNet" model based on deep convolutional neural network for atmospheric visibility prediction [23].

III. Model

In this section, we first formulate our time series forecasting task (section III-A). Then, the details of each component in our proposed network model are presented: i) convolutional component is introduced in section III-B, ii) input attention mechanism based encoder is introduced in section III-C; iii) temporal attention mechanism based decoder is introduced in section III-D; iv) autoregressive component is introduced in section III-E; and v) summing component is introduced in section III-F. Figure 1 presents the graphical illustration of the proposed model. Note that our proposed model combines the advantages of the Long- and Short-term Time-series network (LSTNet) in [8] and Dual-Stage Attention-Based Recurrent Neural Network (DA-RNN) in [9].

A. Problem Formulation and Notation

In this paper, we aim to develop a neural network model for the task of multi-step ahead multi-variate time series forecasting. Mathematically, given a sequence of fully observed time series $\boldsymbol{Y}_{1:T} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T\}$, where $\boldsymbol{y}_t \in \boldsymbol{R}^n$, n is the number of variables, and T is the length of the input time-series, we aim to predict a series of future time-series signals $\hat{\boldsymbol{Y}}_{T+1:T+h} = \left\{\hat{\boldsymbol{Y}}_{T+1}, \hat{\boldsymbol{Y}}_{T+2}, ..., \hat{\boldsymbol{Y}}_{T+h}\right\}$, where h is the desired horizon ahead of the current timestamp. Here, we formulate the input matrix at timestamp T as $X = \boldsymbol{Y}_{1:T} = \left\{\boldsymbol{y}_1^\top, \boldsymbol{y}_2^\top, ..., \boldsymbol{y}_T^\top\right\} \in \mathcal{R}^{T\times n}$.

B. Convolutional Component

Similar to the LSTNet proposed in [8], we fuse local dependencies for all of the used variables by a 2D convolutional layer, consisting of C filters with width w and height h. Note that the height of the convolutional filters, i.e. h, is set to be the same as number of the utilized variables. The output of the k-filter of the 2D convolutional layer is given by:

$$\boldsymbol{f}_k = \boldsymbol{ReLU}\left(\boldsymbol{W}_k * \boldsymbol{X} + \boldsymbol{b}_k\right), \tag{1}$$

where $*$ denotes the 2-D convolutional operation, and the output f_k is a 1-D vector. The $\boldsymbol{ReLU}$ describes the rectified linear units function. The input matrix X is zero-padded to obtain each vector f_k. Note that the output of the convolutional layer is $\boldsymbol{X}^{conv} = \boldsymbol{f}_{1:C} = \left\{\boldsymbol{f}_1^\top, \boldsymbol{f}_2^\top, ..., \boldsymbol{f}_C^\top\right\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\} \in \mathcal{R}^{T\times C}$.

C. Encoder with Input Attention

The output of the convolutional layer is fed into the encoder part of our network model, which is essentially an recurrent neural network (RNN) that encodes the input sequences into a feature representation, like the task of machine translation [24]–[26]. In this multi-step ahead time-series prediction task, the encoder is applied to learn a mapping from $\boldsymbol{x}_t$ to the hidden states of the encoder, $\boldsymbol{h}_t^{enc}$, at timestamp t, by given the output sequence of the convolutional layer, as: $\boldsymbol{h}_t^{enc} = \boldsymbol{LSTM}\left(\boldsymbol{h}_{t-1}^{enc}, \boldsymbol{x}_t\right)$, where $\boldsymbol{h}_t \in \mathcal{R}^m$ denotes the hidden state of the encoder at time t, m is the size of the hidden state, and $\boldsymbol{LSTM}$ represents the long short-term memory (LSTM) unit [27]. In this work, we utilize LSTM unit instead of gated recurrent unit (GRU) to capture long-term dependencies. At timestamp t, the memory cell with the states $\boldsymbol{s}_t^{enc}$ in each $\boldsymbol{LSTM}$ unit is controlled by three sigmoid gates: i) input gate $\boldsymbol{i}_t^{enc}$, ii) forget gate $\boldsymbol{f}_t^{enc}$, and iii) output gate $\boldsymbol{o}_t^{enc}$. We can formulate the updating process as:

$$\boldsymbol{f}_t^{enc} = \sigma\left(\boldsymbol{W}_f^{enc} \times \text{concat}\left\{\boldsymbol{h}_{t-1}^{enc}; \boldsymbol{x}_t\right\} + \boldsymbol{b}_f^{enc}\right) \tag{2}$$

$$\boldsymbol{i}_t^{enc} = \sigma\left(\boldsymbol{W}_i^{enc} \times \text{concat}\left\{\boldsymbol{h}_{t-1}^{enc}; \boldsymbol{x}_t\right\} + \boldsymbol{b}_i^{enc}\right) \tag{3}$$

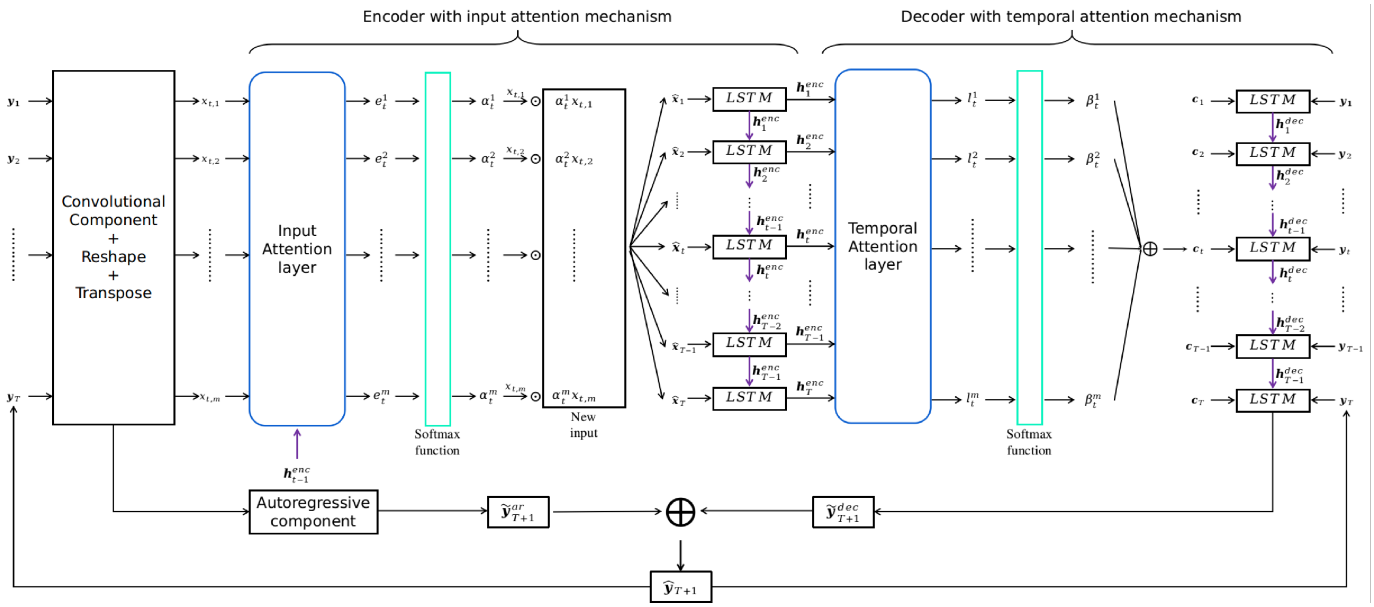
$$\boldsymbol{o}_t^{enc} = \sigma\left(\boldsymbol{W}_o^{enc} \times \text{concat}\left\{\boldsymbol{h}_{t-1}^{enc}; \boldsymbol{x}_t\right\} + \boldsymbol{b}_o^{enc}\right) \tag{4}$$

$$\begin{aligned}\boldsymbol{s}_t^{enc} &= \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1}^{enc} \\ &+ \boldsymbol{i}_t^{enc} \odot \tanh\left(\boldsymbol{W}_s^{enc} \times \text{concat}\left\{\boldsymbol{h}_{t-1}^{enc}; \boldsymbol{x}_t\right\} + \boldsymbol{b}_s^{enc}\right)\end{aligned} \tag{5}$$

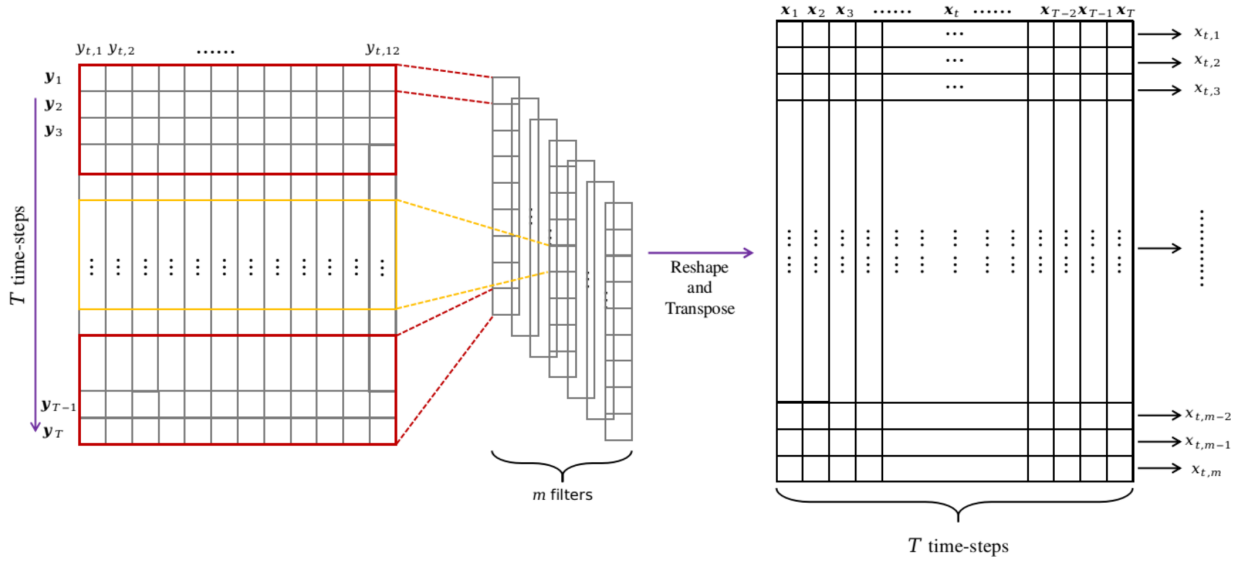
$$\boldsymbol{h}_t^{enc} = \boldsymbol{o}_t^{enc} \odot \tanh\left(\boldsymbol{s}_t\right) \tag{6}$$

where $\text{concat}\{\cdot\}$ represents the concatenate operator, $\text{concat}\{\boldsymbol{h}_{t-1}; \boldsymbol{x}_t\} \in \mathcal{R}^{m+T}$ is the concatenation of the previous hidden state, $\boldsymbol{h}_{t-1}$, and the current input, $\boldsymbol{x}_t$, at the timestamp t, σ denotes the sigmoid function, and $\odot$ is the Hadamard product operator. $\boldsymbol{W}_f$, $\boldsymbol{b}_f$, $\boldsymbol{W}_i$, $\boldsymbol{b}_i$, $\boldsymbol{W}_o$, $\boldsymbol{b}_o$, $\boldsymbol{W}_s$, and $\boldsymbol{b}_s$, are learnable parameters. We use LSTM unit here to overcome the gradient vanishing problem for capturing long-term dependencies, as the cell states in the LSTM units are continuously updating over time.

Next, we follow the idea proposed in [9], which is inspired by the human attention system for elementary features selection [28]. We incorporate the input attention mechanism into our network model to adaptively extract relevant driving series. A feed-forward multi-layer perceptron is used to build up the input attention mechanism for the k-th convolutional feature



(a) Network model



(b) Convolutional component

Fig. 1. Overview of the proposed multi-step LSTM prediction model.

$\boldsymbol{f}_k^\top$, according to the hidden state, i.e., $\boldsymbol{h}_{t-1}$, and the cell state, i.e., $\boldsymbol{s}_{t-1}$, in the previous timestamp in the LSTM unit, as:

$$e_t^k = \boldsymbol{v}_{enc}^\top \tanh\left(\boldsymbol{W}_e \text{concat}\left\{\boldsymbol{h}_{t-1}; \boldsymbol{s}_{t-1}\right\} + \boldsymbol{b}_e + \boldsymbol{U}_e \boldsymbol{f}_k^\top\right) \quad (7)$$

where $\boldsymbol{v}_{enc}$, $\boldsymbol{W}_e$, $\boldsymbol{b}_e$, and $\boldsymbol{U}_e$ are learnable parameters. Note that $\boldsymbol{v}_e$ is a vector of length T, $\boldsymbol{W}_e$ is a 2-D matrix with the size of $T \times 2m$, $\boldsymbol{b}_e$ is a vector of length T, and $\boldsymbol{U}_e$ is a 2-D matrix with the size of $T \times T$. Hence, the adaptively extracted driving series can be mathematically expressed as:

$$\tilde{\boldsymbol{X}} = (\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \cdots, \tilde{\boldsymbol{x}}_T) \quad (8)$$

where $\tilde{\boldsymbol{x}}_t$ is given by:

$$\tilde{\boldsymbol{x}}_t = \left(\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \cdots, \alpha_t^C x_t^C\right) \quad (9)$$

where the attention weights α_t^k is utilized to measure the importance of the k-th convolutional features at timestamp t. They are obtained by applying softmax function on e_t^k as:

$$\alpha_t^k = \frac{\exp\left(e_t^k\right)}{\sum_{i=1}^{C} \exp\left(e_t^i\right)}, \quad (10)$$

Note that we use softmax function here to ensure the summation of all attention weights to be 1. Thus, the hidden state at timestamp t is updated as: $\boldsymbol{h}_t^{enc} = \boldsymbol{LSTM}\left(\boldsymbol{h}_{t-1}^{enc}, \tilde{\boldsymbol{x}}_{t-1}\right)$. Note that here we use the extracted driving series $\tilde{\boldsymbol{x}}_{t-1} \in \mathcal{R}^C$ to adaptively focus on the important part instead of using the

convolutional features $\boldsymbol{x}_{t-1} \in \mathcal{R}^C$ which treat all features equally to update the parameters for the LSTM unit.

D. Decoder with Temporal Attention

Another LSTM unit based RNN is used to decode the encoded information in the previous step. Found by [24], [25], [29], when the input sequences are too long (i.e., length greater than 50), the performance of the encoder-decoder architecture may be seriously degraded. To maintain the prediction accuracy in the long-term part, we introduce the temporal attention mechanism in our network model, as [8], [9], [30], to adaptively select the relevant encoded hidden states across entire input time steps. We determine the attention weights for each encoded hidden state at timestamp t, according to the hidden state and cell state of the LSTM unit for the decoder in the previous timestamp $t-1$, as:

$$l_t^i = \boldsymbol{v}_{dec}^\top \tanh\left(\boldsymbol{W}_{dec}\text{concat}\left\{\boldsymbol{d}_{t-1}; \boldsymbol{s}_{t-1}^{dec}\right\} + \boldsymbol{U}_{dec}\boldsymbol{h}_i\right) \quad (11)$$

where i is an integer $\in [1, T]$, both $\boldsymbol{d}_{t-1}$, and $\boldsymbol{s}_{t-1}^{dec}$ is a vector with length p, representing the hidden state and the cell state of the LSTM unit for the decoder in the timestamp $t-1$, respectively. concat$\{\cdot\}$ represents the concatenate operator. Hence, concat$\left\{\boldsymbol{d}_{t-1}; \boldsymbol{s}_{t-1}^{dec}\right\}$ is a vector with length $2p$. $\boldsymbol{v}_{dec}$, $\boldsymbol{W}_{dec}$, and $\boldsymbol{U}_{dec}$ are learnable parameters. Note that $\boldsymbol{v}_{dec}$ is a vector with length m, $\boldsymbol{W}_{dec}$ is a 2-D matrix with size $m \times 2p$, and $\boldsymbol{U}_{dec}$ is a 2-D matrix with size $m \times m$.

Next, we compute the context vector, $\boldsymbol{c}_t$, by using all hidden states, $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_T\} \in \mathcal{R}^{T \times m}$, of the LSTM unit for the encoder, as: $\boldsymbol{c}_t = \sum_{i=1}^{T} \beta_t^i \boldsymbol{h}_i$, where i is an integer $\in [1, T]$. The attention weights β_t^i are obtained by applying softmax function on l_t^i, as:

$$\beta_t^i = \frac{\exp\left(l_t^i\right)}{\sum_{j=1}^{T} \exp\left(l_t^i\right)} \quad (12)$$

where l_t^i computed from Eq.(11). Note that β_t^i is used to measure the importance of the i-th hidden states for encoding context vector. From Eq.(12), we find that context vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_T$ are varied for different timestamp.

Next, we combine the context vectors with the input time series to form new input time series at each timestamp t, as:

$$\tilde{\boldsymbol{y}}_t = \tilde{\boldsymbol{W}}_t \text{concat}\left\{\boldsymbol{y}_t; \boldsymbol{c}_t\right\} + \tilde{\boldsymbol{b}}_t, \quad (13)$$

where concat$\{\cdot\}$ represents the concatenate operator, concat$\{\boldsymbol{y}_t; \boldsymbol{c}_t\}$ is the concatenation of the input time-series, $\boldsymbol{y}_t$, and the context vectors, $\boldsymbol{c}_t$, at timestamp t. $\tilde{\boldsymbol{W}}_t$ and $\tilde{\boldsymbol{b}}_t$ are learnable parameters. Note that the function of 2-D matrix $\tilde{\boldsymbol{W}}_t$ and variable $\tilde{\boldsymbol{b}}_t$ in Eq.(13) is to map the concatenation of concat$\{\boldsymbol{y}_t; \boldsymbol{c}_t\}$ to fit the size requirement for the decoder input. Herein, $\tilde{\boldsymbol{W}}_t$ is a 2-D matrix with the size of $n \times (m+n)$. concat$\{\boldsymbol{y}_t; \boldsymbol{c}_t\}$ is a vector with the length of $m+n$, and $\tilde{\boldsymbol{b}}_t$ is a vector with the length of n, at timestamp t. The newly formed input time series, $\tilde{\boldsymbol{y}}_t$, which has the same size as the input time series $\boldsymbol{y}_t$ with a length of n, is utilized to update the hidden state for the decoder at timestamp t, as: $\tilde{\boldsymbol{h}}_t^{dec} = \boldsymbol{LSTM}\left(\tilde{\boldsymbol{h}}_{t-1}^{dec}, \tilde{\boldsymbol{y}}_{t-1}\right)$. The memory cell with the states $\boldsymbol{s}_t^{dec}$ in the LSTM unit [9], [27] is also controlled by three sigmoid gates: i) input gate $\boldsymbol{i}_t^{dec}$, ii) forget gate $\boldsymbol{f}_t^{dec}$, and iii) output gate $\boldsymbol{o}_t^{dec}$ for the decoder to capturing long-term dependencies. We can formulate the updating process of the LSTM unit in our temporal attention based decoder, as:

$$\boldsymbol{i}_t^{dec} = \sigma\left(\boldsymbol{W}_i^{dec} \times \text{concat}\left\{\tilde{\boldsymbol{h}}_{t-1}^{dec}; \tilde{\boldsymbol{y}}_{t-1}\right\} + \tilde{\boldsymbol{b}}_i^{dec}\right) \quad (14)$$

$$\boldsymbol{f}_t^{dec} = \sigma\left(\tilde{\boldsymbol{W}}_f^{dec} \times \text{concat}\left\{\tilde{\boldsymbol{h}}_{t-1}^{dec}; \tilde{\boldsymbol{y}}_{t-1}\right\} + \tilde{\boldsymbol{b}}_f^{dec}\right) \quad (15)$$

$$\boldsymbol{o}_t^{dec} = \sigma\left(\boldsymbol{W}_o^{dec} \times \text{concat}\left\{\tilde{\boldsymbol{h}}_{t-1}^{dec}; \tilde{\boldsymbol{y}}_{t-1}\right\} + \tilde{\boldsymbol{b}}_o^{dec}\right) \quad (16)$$

$$\begin{aligned}\tilde{\boldsymbol{s}}_t^{dec} &= \boldsymbol{f}_t^{dec} \odot \tilde{\boldsymbol{s}}_{t-1}^{dec} \\ &+ \boldsymbol{i}_t^{\boldsymbol{dec}} \odot \tanh\left(\boldsymbol{W}_s^{dec} \times \text{concat}\left\{\tilde{\boldsymbol{h}}_{t-1}^{dec}; \tilde{\boldsymbol{y}}_{t-1}\right\} + \tilde{\boldsymbol{b}}_s^{dec}\right)\end{aligned} \quad (17)$$

$$\tilde{\boldsymbol{h}}_t^{dec} = \boldsymbol{o}_t^{dec} \odot \tanh\left(\tilde{\boldsymbol{s}}_t\right) \quad (18)$$

where m^{dec} is the number of hidden units of the $\boldsymbol{LSTM}$ in our decoder, concat$\{\cdot\}$ represents the concatenate operator, concat$\left\{\tilde{\boldsymbol{h}}_{t-1}^{dec}; \tilde{\boldsymbol{y}}_{t-1}\right\} \in \mathcal{R}^{m+n}$ is the concatenation of hidden state of the $\boldsymbol{LSTM}$ unit, $\tilde{\boldsymbol{h}}_{t-1}^{dec}$, for the decoder, and newly formed input time-series, $\tilde{\boldsymbol{y}}_{t-1}$, in the previous timestamp $t-1$. σ denotes the sigmoid function, and $\odot$ is the Hadamard product operator. $\tilde{\boldsymbol{W}}_i^{dec}$, $\tilde{\boldsymbol{b}}_i^{dec}$, $\tilde{\boldsymbol{W}}_f^{dec}$, $\tilde{\boldsymbol{b}}_f^{dec}$, $\tilde{\boldsymbol{W}}_o^{dec}$, $\tilde{\boldsymbol{b}}_o^{dec}$, $\tilde{\boldsymbol{W}}_s^{dec}$, and $\tilde{\boldsymbol{b}}_s^{dec}$ are learnable parameters. $\tilde{\boldsymbol{W}}_i^{dec}$ in Eq.(14), $\tilde{\boldsymbol{W}}_f^{dec}$ in Eq.(15), $\tilde{\boldsymbol{W}}_o^{dec}$ in Eq.(16), and $\tilde{\boldsymbol{W}}_s^{dec}$ in Eq.(17) are 2-D matrices with the size $m^{dec} \times \left(m^{dec} + n\right)$. $\tilde{\boldsymbol{b}}_i^{dec}$ in Eq.(14), $\tilde{\boldsymbol{b}}_f^{dec}$ in Eq.(15), $\tilde{\boldsymbol{b}}_o^{dec}$ in Eq.(16), and $\tilde{\boldsymbol{b}}_s^{dec}$ in Eq.(17) are vectors with the length of p.

We can then obtain the prediction for the next timestamp, i.e., $t = T+1$, as:

$$\hat{\boldsymbol{y}}_{T+1}^{dec} = \boldsymbol{v}_y^\top\left(\boldsymbol{W}_y \times \text{concat}\left\{\hat{\boldsymbol{h}}_t^{dec}; \boldsymbol{c}_t\right\} + \tilde{\boldsymbol{b}}_y^{dec}\right) + \tilde{\boldsymbol{b}}_v^{dec} \quad (19)$$

where $\hat{\boldsymbol{y}}_{T+1}^{dec}$ is the prediction result at timestamp $T+1$ for the encoder-decoder part. $\boldsymbol{W}_y, \tilde{\boldsymbol{b}}_y^{dec}, \tilde{\boldsymbol{b}}_v^{dec}$ are learnable parameters.

E. Autoregressive Component

The scale of input is not sensitive enough in our network model, due to the non-linear nature for both the convolutional components (section III-B) and $\boldsymbol{LSTM}$-based encoder-decoder architecture (sections III-C and III-D). To address this issue, we incorporate the classical autoregressive(AR) model [31] as an extra linear component (shown in Fig.) into our network model. The prediction of the autoregressive(AR) model can be formulated as:

$$\tilde{\boldsymbol{y}}_{t,k}^{ar} = \sum_{jj=0}^{w^{ar}-1} \boldsymbol{W}_{jj}^{ar} \times \boldsymbol{y}_{t-jj,k} + \boldsymbol{b}^{ar} \quad (20)$$

where w^{ar} denotes the window size. $\boldsymbol{W}^{ar}$, and $\boldsymbol{b}^{ar}$ are learnable parameters.

F. Summing Component

The final prediction of our network model at timestamp t is obtained by summing the outputs of the decoder part and the autoregressive component, as:

$$\tilde{\boldsymbol{y}}_{T+1} = \tilde{\boldsymbol{y}}^{dec}_{T+1} + \tilde{\boldsymbol{y}}^{ar}_{T+1} \tag{21}$$

where $\tilde{\boldsymbol{y}}^{dec}_{T+1}$ is determined from Eq.(19), and $\boldsymbol{y}^{ar}_{T+1}$ is determined from Eq.(20). $\hat{\boldsymbol{y}}_{T+1}$ denotes the final prediction of our network model at timestamp $T+1$.

IV. Experiments

In this section, we first describe our customized data sets for the visibility prediction task. Details for data preparation and data preprocessing will be presented in section IV-A. Then, we introduce the parameter settings, objective functions, and optimization strategy for our multi-step LSTM prediction model in section IV-B. Subsequently, we introduce the evaluation metrics to measure the effectiveness of our proposed networks model for visibility prediction in section IV-C. Finally, the prediction results are presented in section IV-D.

A. Data Sets

Data preparation: The data set used in this work is collected at monitoring stations in Shanghai Pudong international airport $(31°8'36"\mathbf{N}, 121°48'19"\mathbf{E})$. Of the 50 collected terms, 12 have been used in this paper for visibility prediction (summarized in the Table I), i.e., 1 min average runway visual range (RVR-1-AVG), 10 min average runway visual range (RVR-10-AVG), 1 min average meteorological optical range (MOR-1-AVG), 10 min average meteorological optical range (MOR-10-AVG), 2 min average wind speed (WS-2-AVG), 10 min average wind speed (WS-10-AVG), 10 min maximum wind speed (WS-10-MAX), query normal height (QNH), query field elevation (QFE), temperature (TP), relative humidity (RD), and dew point temperature (DT).

Data preprocessing: Due to the electronic device's noise signal, unexpected intensive local changes are existed in the capture data. Thus we employ moving average scheme [32]–[34] with the window size of 15 to smooth our data set in the preprocessing stage. Fig. 2 presents the preprocessing results.

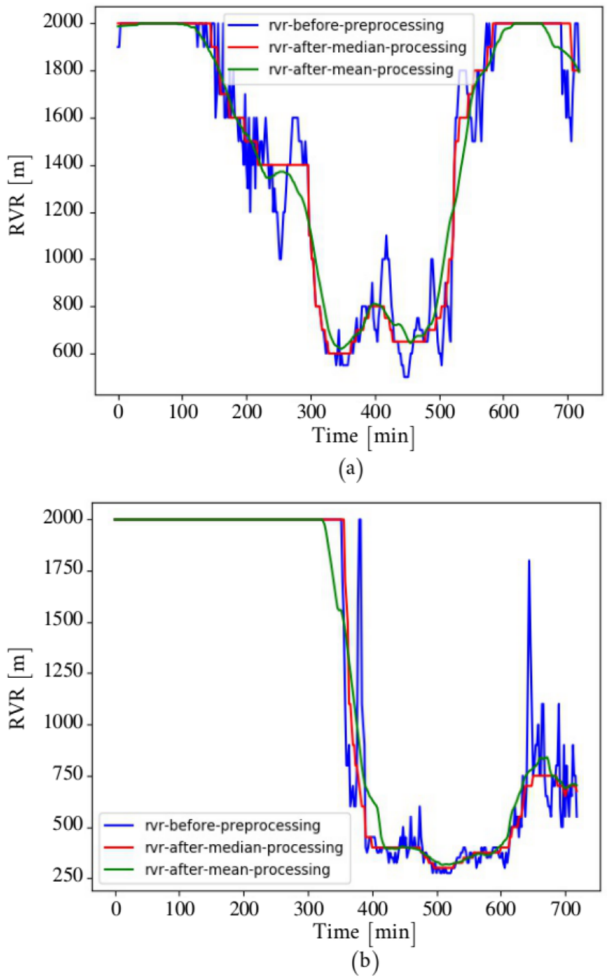


Fig. 2. Data preprocessing. Blue line shows is plot of the RVR without preprocessing; red line is the plot of the preprocessed RVR with median encoding; green line is the plot of the preprocessed RVR with mean encoding.

B. Optimization and Parameter Settings

We simply use absolute loss (L1-loss) function as our objective function for optimization, as:

$$\boldsymbol{\mathcal{L}_\theta} = \min_{\boldsymbol{\theta}} \left\{ \sum_{t=T+1}^{T+H} \left| \boldsymbol{Y}_t - \hat{\boldsymbol{Y}}_t \right| \right\} \tag{22}$$

where $\boldsymbol{\mathcal{L}}_\theta$ represents the objective loss function, $\boldsymbol{\theta}$ represents the parameter set of our network model, $t \in [T+1, T+2, \cdots, T+H]$ is the timestamps set, $\hat{\boldsymbol{Y}}_t$ is the predicted results, while $\boldsymbol{Y}_t$ is the ground-truth at the timestamp t. Note that we do not employ squared error (L2-loss) as the loss function, differing from many previous works [9], [35], [36] for time-series forecasting. The advantage of using L1-loss instead of L2-loss is that it is more robust to the anomalous data in the real time-series data [8] for our visibility forecasting task.

In our work, the optimization strategy is similar to the traditional time series forecasting model [37]–[40]. As the proposed end-to-end network model is differentiable, we can simply employ stochastic gradient descent [41] together with the Adam optimizer [42] to minimize the objective loss function, $\boldsymbol{\mathcal{L}}_\theta$, via back-propagation [43]. We train the network model for 200 epochs on the NVIDIA GeForce RTX 2080 graphics card with the learning rate of 0.001. In each epoch, the entire train set is iteratively trained with the batch size of 16. The proposed multi-step LSTM prediction model is implemented via PyTorch [44], [45]. There are only two primary sets of the parameters in the proposed network model: the number of the hidden states, m_{enc}, the layer number of the $\boldsymbol{LSTM}$ unit, l_{enc}, in the encoder part; the number of the hidden states, m_{dec}, the layer number of the $\boldsymbol{LSTM}$ unit, l_{dec}, in the decoder part. Note that we simply set $m_{dec} = m_{enc}$, and $l_{dec} = l_{enc}$ in the grid search over $m_{dec} = m_{enc} \in \{32, 64, 128, 256, 512\}$, and $l_{dec} = l_{enc} \in \{1, 2, 3, 4\}$. We find

TABLE I
USED VARIABLES

Physics variables	Unit	Abbreviation
1 min average runway visual range	$[m]$	RVR-1-AVG
10 min average runway visual range	$[m]$	RVR-10-AVG
1 min average meteorological optical range	$[m]$	MOR-1-AVG
10 min average meteorological optical range	$[m]$	MOR-10-AVG
2 min average wind speed	$[m/sec]$	WS-2-MAX
10 min average wind speed	$[m/sec]$	WS-10-AVG
10 min maximum wind speed	$[m/sec]$	WS-10-MAX
query normal height	hPa	QNH
query field elevation	hPa	QFE
temperature	$[K]$	TP
relative humidity	$[\%]$	RD
dew point temperature	$[K]$	DT

that $m_{dec} = m_{enc} = 512$, and $l_{dec} = l_{enc} \in \{2\}$ achieve the best performance over the validation set.

C. Evaluation Metrics

We employ mean classification error (MCE) and mean absolute error (MAE) as the evaluation metrics [46] to measure the performance for different methods in this time-series prediction task. i) MCE is defined by:

$$MCE = \sum_{\gamma \in \Omega_{test}} \frac{1}{H} \left\{ \sum_{t=T+1}^{T+H} \mathbb{1}\left\{ \boldsymbol{y}_t^c, \hat{\boldsymbol{y}}_t^c \right\} \right\} \tag{23}$$

where $\mathbb{1}\{\}$ denotes the indicator function , $\boldsymbol{y}_t^c$ represents the ground-truth class, Ω_{test} represents the test set, $\hat{\boldsymbol{y}}_t^c$ represents the predicted class. ii) mean absolute error (MAE) is defined by:

$$MAE = \sum_{\gamma \in \Omega_{test}} \left\{ \left(\sum_{t=T+1}^{T+H} |\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t| \right) \right\} \tag{24}$$

where Ω_{test} represents the test set, $\boldsymbol{y}_t$ is the ground-truth time-series in the timestamp $t \in [T+1, T+h]$, and $\hat{\boldsymbol{y}}_t$ is the predicted time-series in the timestamp $t \in [T+1, T+h]$.

D. Results

To demonstrate the effectiveness of the proposed multi-step LSTM model, we compare it against the LSTNet [8] and DA-RNN [9]. The prediction results of runway visual range are shown in Fig. 3. Our multi-step prediction model achieve the mean MCE of 68.9%, and mean MAE of 306.2 m, slightly better than the results of LSTNet with the mean MCE of 65.4%, and mean MAE of 356.9 m. The prediction results for the intervals of [0, 150), [150, 350), [350, 600), [600, 800), [800, 1500), and [1500, 3000) are given in Table II. The plots of the predictions results of our multi-step LSTM prediction model and LSTNet are presented in Fig. 3.

V. Conclusion

In this paper, we present a deep learning framework for visibility prediction. We reformulate it as a multi-variate time series forecasting task. By introducing the convolutional component into the encoder-decoder architecture with attention mechanism, the proposed network model combines the advantages of LSTNet [8] and DA-RNN [9], [10], and achieves the state-of-the-art prediction accuracy (68.9%) on the runway visual range prediction for the visibility measurement for the airport.

As for future research, there are several promising directions for optimizing this work. First, introduce the skip-connection for long-term prediction. Second, replace the current RNN-based encoder-decoder architecture with the transformer architecture may further improve the prediction accuracy.

References

[1] M. Tudor, "Impact of horizontal diffusion, radiation and cloudiness parameterization schemes on fog forecasting in valleys," *Meteorology and atmospheric physics*, vol. 108, no. 1-2, pp. 57–70, 2010.

[2] C. Lin, Z. Zhang, Z. Pu, and F. Wang, "Numerical simulations of an advection fog event over shanghai pudong international airport with the wrf model," *Journal of Meteorological Research*, vol. 31, no. 5, pp. 874–889, 2017.

[3] A. Grover, A. Kapoor, and E. Horvitz, "A deep hybrid model for weather forecasting," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 379–386. [Online]. Available: https://doi.org/10.1145/2783258.2783275

[4] D. Dutta and S. Chaudhuri, "Nowcasting visibility during wintertime fog over the airport of a metropolis of india: decision tree algorithm and artificial neural network approach," *Natural Hazards*, vol. 75, no. 2, pp. 1349–1368, 2015.

[5] I. Bartoková, A. Bott, J. Bartok, and M. Gera, "Fog prediction for road traffic safety in a coastal desert region: Improvement of nowcasting skills by the machine-learning approach," *Boundary-layer meteorology*, vol. 157, no. 3, pp. 501–516, 2015.

[6] J. B. Bremnes and S. C. Michaelides, "Probabilistic visibility forecasting using neural networks," in *Fog and Boundary Layer Clouds: Fog Visibility and Forecasting*. Springer, 2007, pp. 1365–1381.

[7] C. Marzban, S. Leyton, and B. Colman, "Ceiling and visibility forecasts via neural networks," *Weather and forecasting*, vol. 22, no. 3, pp. 466–479, 2007.

[8] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 2018, pp. 95–104.

[9] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 2627–2633.

[10] Y. Liu, C. Gong, L. Yang, and Y. Chen, "Dstp-rnn: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," *Expert Systems with Applications*, vol. 143, p. 113082, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419307997

TABLE II
RVR PREDICTION PERFORMANCE

Parameters	0-150	150-350	350-600	600-800	800-1500	1500-3000	mean
RVR MCE (multi-step LSTM) [%]	7.5	12.9	18.9	11.2	25.3	92.4	68.9
RVR MCE (LSTNet) [%]	7.1	11.7	17.9	11.1	25.3	91.9	65.4
RVR MAE (multi-step LSTM) [m]	1023.9	816.7	619.2	865.2	568.9	119.8	306.2
RVR MAE (LSTNet) [m]	1032.1	836.7	627.9	856.1	625.3	136.7	356.9

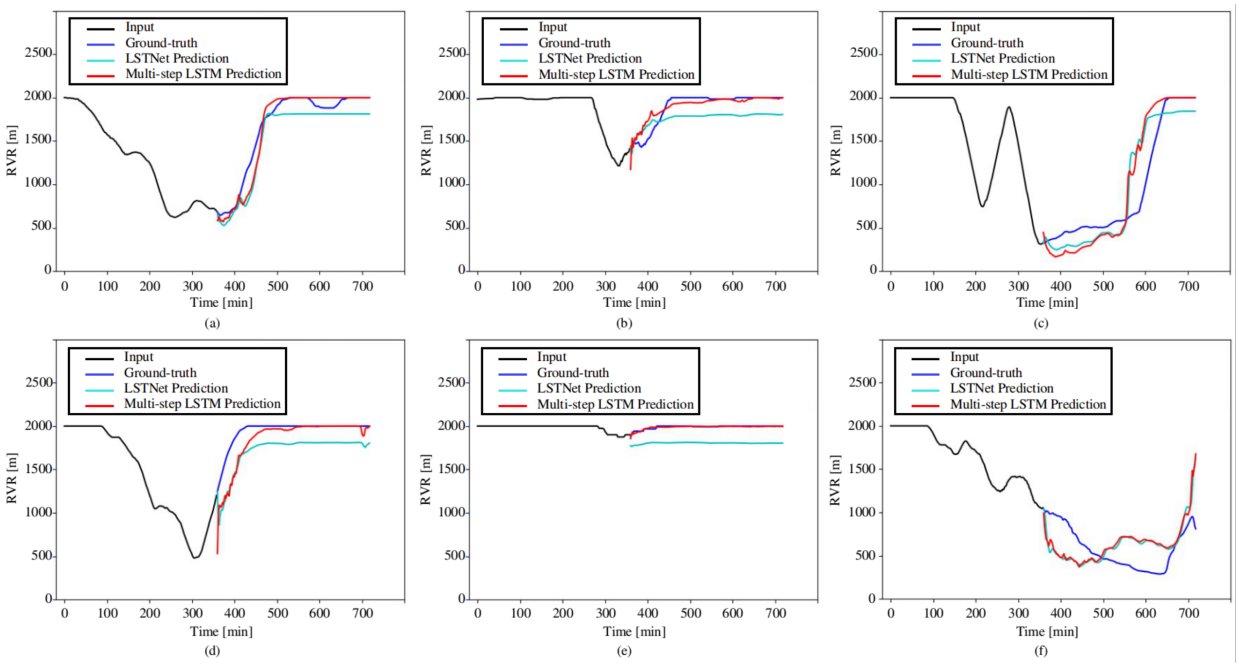


Fig. 3. Plots of the predictions results of our proposed multi-step LSTM prediction model (read line) and LSTNet (cyan line) and the monitored ground-truth (blue line).

[11] T. Bergot and D. Guedalia, "Numerical forecasting of radiation fog. part i: Numerical model and sensitivity tests," *Monthly Weather Review*, vol. 122, no. 6, pp. 1218–1230, 1994.

[12] P. Goswami and S. Sarkar, "An analogue dynamical model for forecasting fog-induced visibility: validation over delhi," *Meteorological Applications*, vol. 24, no. 3, pp. 360–375, 2017.

[13] M. Shatunova, G. Rivin, and I. Rozinkina, "Visibility forecasting for february 16–18, 2014 for the region of the sochi-2014 olympic games using the high-resolution cosmo-ru1 model," *Russian Meteorology and Hydrology*, vol. 40, no. 8, pp. 523–530, 2015.

[14] G. Steeneveld, R. Ronda, and A. Holtslag, "The challenge of forecasting the onset and development of radiation fog using mesoscale atmospheric models," *Boundary-Layer Meteorology*, vol. 154, no. 2, pp. 265–289, 2015.

[15] B. Zhou, J. Du, I. Gultepe, and G. Dimego, "Forecast of low visibility and fog from ncep: Current status and efforts," *Pure and Applied Geophysics*, vol. 169, no. 5-6, pp. 895–909, 2012.

[16] S. J. Dietz, P. Kneringer, G. J. Mayr, and A. Zeileis, "Forecasting low-visibility procedure states with tree-based statistical methods," *Pure and Applied Geophysics*, vol. 176, no. 6, p. 2631–2644, 2019.

[17] P. Kneringer, S. J. Dietz, G. J. Mayr, and A. Zeileis, "Probabilistic nowcasting of low-visibility procedure states at vienna international airport during cold season," *Pure and Applied Geophysics*, vol. 176, no. 5, pp. 2165–2177, 2019.

[18] L. Ortega, L. D. Otero, and C. Otero, "Application of machine learning algorithms for visibility classification," in *2019 IEEE International Systems Conference (SysCon)*. IEEE, 2019, pp. 1–5.

[19] D. Bari, "Visibility prediction based on kilometric nwp model outputs using machine-learning regression," in *2018 IEEE 14th International Conference on e-Science (e-Science)*. IEEE, 2018, pp. 278–278.

[20] W. R. Ryerson and J. P. Hacker, "A nonparametric ensemble postprocessing approach for short-range visibility predictions in data-sparse areas," *Weather and Forecasting*, vol. 33, no. 3, pp. 835–855, 2018.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[22] L. Zhu, G. Zhu, L. Han, and N. Wang, "The application of deep learning in airport visibility forecast," *Atmospheric and Climate Sciences*, vol. 7, no. 03, p. 314, 2017.

[23] A. Palvanov and Y. I. Cho, "Visnet: Deep convolutional neural networks for forecasting atmospheric visibility," *Sensors*, vol. 19, no. 6, p. 1343, 2019.

[24] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[25] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[26] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association

for Computational Linguistics, Oct. 2014, pp. 103–111. [Online]. Available: https://www.aclweb.org/anthology/W14-4012
[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[28] R. Hübner, M. Steinhauser, and C. Lehle, "A dual-stage two-phase model of selective attention." *Psychological review*, vol. 117, no. 3, p. 759, 2010.
[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[30] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "Geoman: Multi-level attention networks for geo-sensory time series prediction," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3428–3434. [Online]. Available: https://doi.org/10.24963/ijcai.2018/476
[31] M. Nerlove and F. X. Diebold, "Autoregressive and moving-average time-series processes," in *Time Series and Statistics*. Springer, 1990, pp. 25–35.
[32] R. A. Johnson, D. W. Wichern *et al.*, *Applied multivariate statistical analysis*. Prentice hall Upper Saddle River, NJ, 2002, vol. 5, no. 8.
[33] S. V. Crowder, "Design of exponentially weighted moving average schemes," *Journal of Quality Technology*, vol. 21, no. 3, pp. 155–162, 1989.
[34] J. S. Hunter, "The exponentially weighted moving average," *Journal of quality technology*, vol. 18, no. 4, pp. 203–210, 1986.
[35] Y. Liu, C. Gong, L. Yang, and Y. Chen, "Dstp-rnn: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," *Expert Systems with Applications*, vol. 143, p. 113082, 2020.
[36] Z. Mariet and V. Kuznetsov, "Foundations of sequence-to-sequence modeling for time series," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 408–417. [Online]. Available: http://proceedings.mlr.press/v89/mariet19a.html
[37] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.
[38] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3504–3512. [Online]. Available: http://papers.nips.cc/paper/6321-retain-an-interpretable-predictive-model-for-healthcare-using-reverse-time-attention-mechanism.pdf
[39] T. Guo, T. Lin, and N. Antulov-Fantulin, "Exploring interpretable LSTM neural networks over multi-variable data," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 2494–2504. [Online]. Available: http://proceedings.mlr.press/v97/guo19b.html
[40] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.
[41] L. Bottou, "Stochastic learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 146–168.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Advances in Neural Information Processing Systems Workshop*, 2017.
[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
[46] M. Plutowski, G. Cottrell, and H. White, "Experience with selecting exemplars from clean data," *Neural Networks*, vol. 9, no. 2, pp. 273 – 294, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608095000992