IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 12, 2021, accepted November 22, 2021, date of publication November 30, 2021,
date of current version December 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3131533

A Comparative Analysis of Deep Neural Networks for Hourly Temperature Forecasting

EHTASHAMUL HAQUE[1], SANZANA TABASSUM[1], (Student Member, IEEE), AND EKLAS HOSSAIN[2], (Senior Member, IEEE)

[1]Department of Electrical and Electronic Engineering, Islamic University of Technology, Gazipur 1704, Bangladesh
[2]Department of Electrical Engineering and Renewable Energy, Oregon Renewable Energy Center (OREC), Oregon Tech, Klamath Falls, OR 97601, USA

Corresponding author: Eklas Hossain (eklas.hossain@oit.edu)

ABSTRACT High-resolution temperature forecasting can often prove to be challenging for conventional machine learning models as temperature is highly seasonal and varies with the time of the year as well as with passing hours of the day. In most cases, only the daily extremes or mean temperatures are provided by temperature forecasting methods. However, with the growing availability of data and the development of deep neural networks (DNNs) capable of detecting complex relationships, high-resolution temperature forecasting is becoming easier. Typically, historical temperature data along with multiple meteorological sensor data is used for temperature forecasting which increases the complexity of the system making it harder and costlier to implement physically. In this paper, high-resolution hourly temperature forecasting is performed using only historical temperature data. The paper presents a comparative analysis among four popular DNNs- simple recurrent neural network (SRN), gated recurrent unit (GRU), long-short term memory (LSTM), convolutional neural network (CNN), and two hybrid models- CNN-LSTM parallel network and GRU-LSTM parallel network trained on Beijing temperature dataset. Experimental results showed GRU-LSTM parallel network obtained the lowest RMSE (1.691° C) whereas CNN has the best computational efficiency obtaining a slightly worse RMSE (1.759° C). Additionally, a robustness analysis is performed on temperature data from four additional geographically diverse locations (Toronto, Las Vegas, Seattle, and Dallas) which reveals GRU to be the most consistent algorithm. Finally, the paper establishes a correlation between the model performance and the dataset based on their variance and mean absolute deviation with reference to the training dataset.

INDEX TERMS Deep neural network, CNN, LSTM, CNN-LSTM parallel, temperature forecasting, GRU, RNN, GRU-LSTM parallel, robustness.

I. INTRODUCTION

Temperature forecasting is one of the most consistent areas of research owning to its direct impact on utility demand, living conditions, agriculture, and various industries. Temperature has a high correlation to electric load demand in particular and therefore, temperature forecast is a prerequisite for many load forecasting schemes. These forecasts are usually provided by weathers stations in many countries, but often only predict the daily extremes (maximum and minimum) or average temperatures. Moreover, it does not specify what time of the day this maximum or minimum will occur. The extreme temperatures only help to predict the peak load; however, with the growing availability of data, higher resolution temperature predictions can be made which will aid utilities with the scheduling, supply operation and preparation for sudden load change to a great extent. In this regard, hourly forecast of temperature is an important feature that can further improve the prediction horizon of many other applications.

To better schedule the generation scheme and avoid under-generation or overgeneration, many utilities require hourly temperature data for short-term load forecasting (STLF) [1]. A significant percentage of electric demand comes from heating, ventilation and air conditioning (HVAC) which consumes more than 40% of a building's power on average [2]. HVAC is highly temperature-dependent, and STLF such as 1 hour ahead (h ahead), 2h ahead and 3h ahead can be crucial for preparing the HVAC systems to adapt to

The associate editor coordinating the review of this manuscript and approving it for publication was Grigore Stamatescu.

the change of load and enhance the operational safety of the electric network.

Zhao and Liu [3] proposed a hybrid PLS-SVM model that takes into account meteorological parameters and historical data to perform up to 3h ahead and 24h ahead load forecasting to optimize HVAC operations. The authors showed that accuracy of the hourly temperature forecast directly affects the proposed model. Higher resolution such as 1h ahead, 2h ahead and 3h ahead temperature forecasts yield higher accuracy for the load forecasting model compared to using daily extreme temperature forecasts. Hourly temperature data is also required to analyze test reference years (TRYs) and design summer years (DSYs) for energy use, to calculate plant sizing, and to simulate building performance during hot summers [4].

Shao and Lister [5] proposed a model which predicts the hourly road surface temperature and state (wet/ice/dry) using meteorological data from seven countries. This model is a short-term model that predicts up to 3h ahead which integrates an hourly temperature forecasting scheme as a prerequisite feature for the next stage of the proposed forecasting model. A similar study by Bogren and Gustavsson [6] used hourly air temperature forecast to predict the road surface temperature. In agriculture, Kim *et al.* [7] used hourly air temperature forecasts to estimate the duration of leaf hydration retainability. Hourly temperature can even affect biological parameters, such as the mortality burden of hourly temperature variability which was studied extensively [8]. Another significant application of hourly temperature forecasting is in photovoltaic (PV) generation. For seamless grid integration, predicting hourly fluctuations in PV generation is crucial. Since the output of a PV system is a function of temperature, hourly temperature forecasts are a prerequisite in the solar industry.

So, there are a plethora of applications for hourly temperature forecasting. After addressing the necessity of high resolution hourly forecasts, the discussion proceeds to assess the hourly forecast techniques that have been used so far as well as the state-of-the-art regarding this topic.

II. TEMPERATURE FORECASTING METHODS

Weather forecasting mainly takes one of three routes- traditional physics-based, statistical and NN or DNN models. This section briefly explores the different techniques, their advantages and drawbacks.

A. PHYSICS BASED MODELS

Physics-based weather forecasting is the traditional method and is still used by a number of public weather forecast providers. These methods mainly take into account physical parameters like solar irradiance, wind speed, humidity, precipitation, cloud covers, etc. and use theoretical formulae to calculate the future temperature. Zhao and Liu [3] presented a purely physics-based temperature forecasting model to determine the temperature which is a prerequisite for the load forecasting part of their study. The study used a heat conduction equation that assessed parameters such as heat capacity, conductivity, current temperature, surface albedo, solar irradiance, net longwave irradiance, ground conductive heat flux density, sensible and latent heat flux densities to derive the road surface temperature. Physics-based models require sensor measurements from multiple sources to compute the temperature; moreover, these values vary significantly across different locations. These models tend to work better for daily temperature forecasting rather than short horizon predictions.

B. STATISTICAL MODELS

Mathematical models started gaining momentum around the 1990s. Since the temperature forecasts at that time only provided maximum and minimum temperature without specifying what time of the day it will occur, the hourly electric load curve had to be generated through interpolation of the two extremes. Data-driven weather forecasting models are built using different statistical and machine learning algorithms. Such models can significantly decrease the setup cost by trading off more historical data for additional sensor data. However, these models may require extensive historical data to yield good accuracy. Recently, with the increased availability of precise data, data-driven models for weather forecasting have gained popularity and are actively being studied. Statistical models such as, autoregressive integrated moving average (ARIMA) use time-series analysis to predict long-term change in data like daily and monthly time horizons [9]. ARIMA is one of the most common linear statistical techniques and a form of regression analysis used in time series forecasting. The auto-regressive component of ARIMA regresses some of the lagged data, then integration is performed to make the data stationary, and the moving-average incorporates preceding error terms from a moving average model applied to lagged observations. One of the biggest drawbacks of ARIMA is that it is negatively affected by seasonality, and temperature is a highly seasonal dataset. If stationarity is not confirmed in a trend, computation throughout the whole process might not be accurate [10]. So ARIMAs are not the best choice for temperature forecasting.

C. NEURAL NETWORK MODELS

In recent times, NN models have become increasingly popular specially for short-term predictions such as hourly and daily time horizons compared to long-term predictions achieved through statistical models. Existing research mostly focus on temperature forecasting using consistent time unit data where both the input and target data are of the same time unit, for example, using daily input data to forecast day-ahead temperature. However, with the increased availability of high-resolution data, and continued development of processing units, it is now possible to predict a time frame of different duration compared to the input. Both hourly and daily patterns can be employed to forecast daily temperatures, but as the data is abundant and detailed, it is essential to process them efficiently and accurately. With the correct models,

TABLE 1. Literature on temperature forecasting using statistical and neural network models.

Literature	Method	Time horizon	Year	Remarks
Khotanzad et al. [11]	BP MLP	Up to seven days	1997	Applied BP MLP (backpropagation multilayered perception) for forecasting any horizon up to seven days. However, BP comes with disadvantages such as setting parameter values like learning rate, connection weights, slow convergence and risk of getting trapped in any local minima; moreover, integrating it with time series requires extra preprocessing of data, especially while flattening lag observations into feature vectors [12].
Hippert et al. [13]	Hybrid ANN and ARIMA	Hourly	2000	When sequential forecasting is performed, e.g., 1h ahead temperature is predicted at a time and then the predicted value is used as one of the inputs in the forecasting of the next hour's temperature- ANN might yield chaotic output. The proposed hybrid model claims to mitigate this drawback.
Methaprayoon et al. [14]	Multistage ANN	Hourly	2007	The back-end hourly temperature forecaster of an STLF model performed poorly for temperature forecasting, and it was exclaimed that the temperature forecast error was the major cause behind higher errors in STLF.
V. Vamitha et al. [15]	Fuzzy & Markov chain	Daily average	2012	Through the fuzzification of the temperature data, four categorical data sequences were obtained and a multivariate Markov chain was applied.
Verhelst et al. [2]	ARX, ARMAX	Hourly	2017	These models are multivariate and computation expensive. ANN does not require input data in stationary format but ARIMAX requires stationary data, which adds to the computation costs during pre-processing.
T Tran et al. [16]	ANN, RNN, LSTM	Daily maximum	2020	Presented a comparative study between ANN, RNN and LSTM for different day ahead temperature forecasts upto 35 days after dividing the forecasts into four different seasons.
Z Zhang et al. [17]	CRNN	Daily average	2020	This paper presented a way for geographical temperature forecasting. They predicted the next day average temperature of 800 temperature stations and adjacent areas of China using one single model.
D. Kreuzer et al. [18]	Naive, SARIMA, LSTM, convLSTM	Hourly	2020	Proposed a convLSTM network using relative humidity, cloud coverage, hourly precipitation, wind speed, relative air pressure and wind direction along with air temperature as input. The increased input parameters yield improved accuracy than univariate inputs but also come with higher computation cost.
Lee et al. [19]	MLP, LSTM, CNN	Daily extreme	2020	The latest addition to the DNN based short-term temperature forecasting studies, which presented a comparative analysis between MLP, LSTM and CNN, but only predicted the extremes and average daily temperature.
T. Toharudin et al. [20]	LSTM, Facebook Prophet model	Daily extremes	2021	Proposed that LSTM and Prophet's model can mitigate RNN's inability to handle data with longer time spans by adding a cell state that retains previous information.

hourly temperature data can even be used to predict the hourly temperature of the next day to a limit before the errors become too significant. In this context, NN models have powerful versatility to process large amounts of more detailed data, which this paper aims to present.

Existing research on temperature forecasting using statistical and NNs are tabulated in Table 1. It can be observed that, earlier versions of temperature forecasting use different statistical models such as MLP, ARIMA or modified ARIMAs. Some of these papers include hourly temperature forecasting as the prerequisite of a load forecasting model [12]. More recent works started adopting NNs and DNNs that yield higher accuracy compared to statistical models, which is discussed in [18]. However, most of these papers use NNs to predict daily extremes and average [19]. To the best of the authors knowledge, only one forecasting model predicts hourly horizon using DNN, achieving an hourly average RMSE value of 2.10 using their proposed convLSTM model tested on a temperature dataset of Germany. However, it uses five meteorological parameters as input. This not only increases computation cost, but requires expensive sensor data as well [18]. Univariate regression using NNs can mitigate this drawback. In addition, temperature patterns differ significantly based on geographical location, so it will be interesting to observe how DNNs trained on a local temperature pattern performs on a different region. It is apparent that a study comparing the performance of the most recent DNNs for hourly temperature forecasting, taking into account spatial diversity (local and geographically diverse) and robustness is yet to be explored.

This study intends to address the existing research gap and make the following significant contributions:

- Comparative analysis for hourly temperature forecasting using four of the most popular DNNs (SRN, LSTM, GRU, CNN) and two hybrid DNNs (CNN-LSTM parallel, GRU-LSTM parallel), with univariate time series data.
- Comparison of hour-by-hour prediction and single run prediction. In addition, explore the effect of normalization of input data.
- A robustness analysis to check if the DNNs perform similarly for four different regions and input patterns, thus grading their ability to generalize.
- Correlation between model performance and different input patterns based on variance and mean absolute deviation (MAD) of the dataset.

The outcome of this study will be especially helpful to determine which DNN might perform best for applications that require hourly temperature forecasting, particularly load forecasting along with other applications mentioned in Section I. The rest of the paper is organized as follows- Section III gives a mathematical and illustrated overview of the DNNs considered in this study. Section IV breaks down the methodology and implementation of the models. Section V presents the outcome of the comparative analysis and Section VI discusses the robustness analysis of the models along with its correlation to different parameters of a dataset. Finally, Section VII concludes the paper with an indication of future scopes.

III. FORECASTING ARCHITECTURE

A. SIMPLE RECURRENT NEURAL NETWORK (SRN)

Conventional feed-forward NNs are ineffective for prediction using sequential data because it assumes all the units of input vector to be independent of time [21]. RNNs differ from conventional feed-forward NNs as they are sequence-based models that allow the learning of time-based dependencies. RNNs have the ability to create temporal correlation from past data with the present state [22]. RNN allows the signal to move forward and backward, and can make a loop in the NN. Thus RNNs work specially well on sequential data where the decision made at the previous time step $(t-1)$ is preserved and utilized on the decision made at the current time step t. SRN is the simplest form of RNN that takes two inputs- current state x_t at time t and previous hidden state h_{t-1}, and updates the values by a non-linear activation function. The recurrent unit has a single hyperbolic tangent (*tanh*) layer. The repeating module of SRN [23] can be expressed by the following equation-

$$h_t = \tanh\left(w_c \cdot [h_{t-1}, x_t] + b\right) \tag{1}$$

where h_t is the hidden neuron at time t, o_t is the output vector and b is the bias value.

Figure 1 illustrates a basic SRN unit. The main drawback of SRN is that it sometimes fails to converge to the optimum minima due to its vanishing gradient problem that might arise during back propagation [24]. So over the course of time, multiple modified versions of RNN have been proposed, some of which have become very popular such as LSTM and GRU.

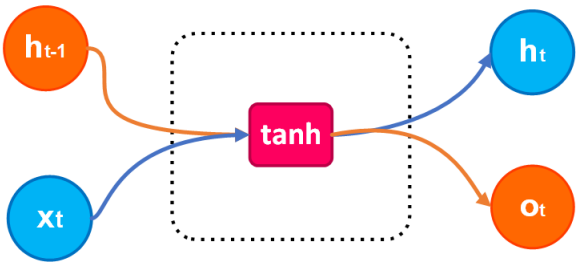

FIGURE 1. Structure of a simple RNN (SRN) cell.

B. LONG SHORT TERM MEMORY (LSTM)

LSTM is a modified version of RNN first proposed by Hochreiter and Schmidhuber [25] which was proposed to mitigate the vanishing gradient problem of SRNs. LSTM can store previous data in its memory unit and add/discard information during the learning process. LSTM has proven to be very effective for sequential data such as signals, protein patterns, text data, time series forecasting etc. Instead of the single hyperbolic tangent layer in the recurrent unit of SRN, LSTM has four layers. The basic components of an LSTM unit are- a memory cell and three gating units- input gate (i_t), output gate (o_t) and forget gate (f_t) which are shared by all cells in the block. In total, there are three inputs and two outputs. Each layer receives an input x_t, previous hidden layer state h_{t-1} and previous cell state c_{t-1}. The hidden layer derives a hidden state vector h_t and the output cell state c_t. The purpose of the input gate is to determine if a cell c_t should be updated by x_t or not, f_t decides if the previous cell c_{t-1} should be forgotten, and the output of h_t depends on o_t to control which part of c_t should be used. An activation function normalizes the state of the gates, 0 indicating no information flow and 1 indicating full flow of information through the gate. The basic structure of an LSTM unit is illustrated in Figure 2.

The nodal outputs of a LSTM network are computed as follows [26]:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2}$$

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1,} x_t\right] + b_f\right) \tag{3}$$

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1,} x_t\right] + b_c\right) \tag{4}$$

$$C_t = \mathrm{f_t} * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t]\right) + b_0 \tag{6}$$

$$h_t = o_t * \tanh\left(c_t\right) \tag{7}$$

where input variable at time step t is denoted by x_t. c_t and h_t are cell state and hidden state respectively. $\tilde{c}_t$ is referred to as the candidate cell calculated in Eq.4 whose output through the *tanh* function has a value between -1 and 1. W_f, W_i, W_c, W_o denote different weight matrices for input vectors. The σ represents the sigmoid activation function and $*$ symbol

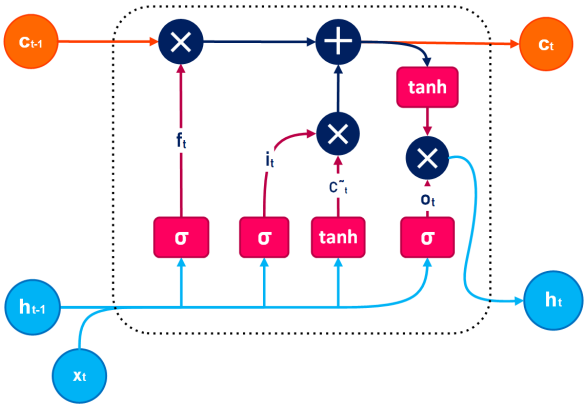

FIGURE 2. Structure of a basic long-short term memory (LSTM) cell.

denotes element-wise multiplication operation. Lastly, b_i, b_c, b_f and b_0 refer to the bias values of the i_t, c_t, f_t and o_t respectively. $\tilde{C}_t$ stores the state information and is updated by Eq. 7. Eq. 4 and 7 uses a hyperbolic tangent operator to calculate the memory cell and o_t. This enables the LSTM network to retain the useful information across different timescales. LSTM was modified to avoid the vanishing gradient problem by allowing gradients to flow unchanged. However, LSTM networks are still vulnerable to the exploding gradient problem [27].

C. GATED RECURRENT UNIT (GRU)

Another popular modification of the RNN is the GRU proposed by Cho *et al.* [28] with an aim to make the recurrent units adapt and capture the dependencies of different timescales and sequences. The updated mechanism allows the GRU to capture long-term dependencies. A GRU unit encompasses two gates, the reset gate r_t and the update gate z_t. The update gate is similar to the forget gate and input gate

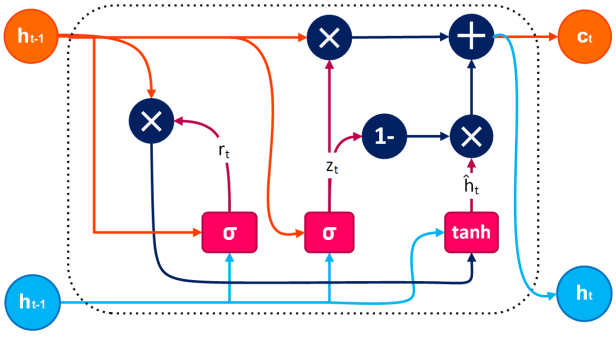

FIGURE 3. Structure of a basic gated recurrent unit (GRU) cell.

in LSTM as it controls storing or erasing potential features from the previous state that can be useful later. Meanwhile, the reset gate controls the amount of information that should be discarded. The reset gate mechanism helps the efficiency of GRU model capacity by allowing it to reset features that are detected to no longer be useful. The basic unit of a GRU is illustrated in Figure 3.

The equations for the input and output of a GRU model are:

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t] + b_z\right) \tag{8}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t] + b_r\right) \tag{9}$$

$$\tilde{h}_t = \tanh\left(W_h \cdot [r * h_{t-1}, x_t] + b_h\right) \tag{10}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t \tag{11}$$

where $\tilde{h}_t$ and h_t are the candidate activation and hidden state at time t, respectively. W_z, W_r, W_h are the weight matrices of update gates, reset gates and hidden states respectively. The "*" is used to express element-wise multiplication and σ is the sigmoid activation function. GRU is an updated version of LSTM that has two gating units that hold the flow of information but it does not have a separate memory cell. As LSTM contains 12 parameters for each separate unit, a fully connected LSTM layer becomes computationally costly to implement, thus GRUs improve the computational efficiency by combining two LSTM gates (the input and forget gates) into a single update gate [22] which might compromise performance a little, but its improved training time makes GRU faster than LSTM.

D. CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN has become a standard, go-to model for computer vision and image classification applications. CNN models are capable of filtering and extracting complex patterns and features from massive visual datasets (with ground-truth labels). It works by automatically learning a large number of filters in parallel specific to a training dataset and repeatedly applying the same filter to an input which results in activations known as a feature map. The fundamental concept utilizes the mathematical operator called convolution to transform two functions into a single function. Convolution can be performed on two functions at a time, but CNN is used up to 4D spatio-temporal processing [29].

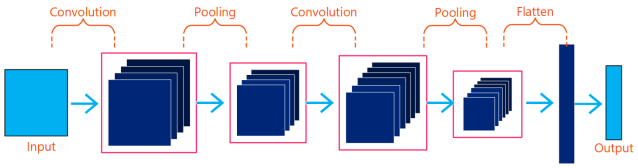

FIGURE 4. Structure of a general convolutional neural network.

However, there was uncertainty regarding how CNN will perform on 1D time series data, specially when the dataset is not sufficient [30]. In the particular case of a 1D convolutional layer, 1D pooling layers are used to create CNNs for signal analysis as well as time series analysis. The internal structure of CNN encompasses three layers- convolutional layer, dense layer and pooling layer. The convolution layers perform convolution operation with the help of linear activation to extract the local features. The forward and back propagation are detailed in the following equations [30]:

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D(w_{ik}^{l-1}, s_i^{l-1}) \tag{12}$$

where w_{ik}^{l-1} denotes the kernel between i^{th} neuron at layer l-1 and the k^{th} neuron at layer l. s_i^{l-1} is the output from the i^{th}

neuron at layer $l-1$. x_k^l and b_k^l are the input and the bias of the k^{th} neuron at layer l, respectively. In order to perform 1D convolution without zero padding, the $conv1D(.,.)$ function was used. This implies that the dimension of s_i^{l-1} (output arrays) are higher than the dimension of x_k^l (input arrays). The intermediate output y_k^l is obtained by applying an activation function $f(.)$ on the input x_k^l using the following equation:

$$y_l^k = f(x_l^k) \;\; and \;\; s_l^k = y_l^k \downarrow ss \tag{13}$$

where $\downarrow ss$ denotes a down-sampling operation with a scalar factor, ss [30]. Down-sampling of the feature map is performed in this layer which reduces several values into one value keeping the integrity of the input data unchanged [19]. The last layer is the dense layer which receives the flattened data of the pooling stage and makes it a 1D output sequence. An attractive feature of 1D CNN is that low-cost hardware implementation is possible as 1D CNNs only perform 1D convolutions, which is basically additions and scalar multiplications. A basic internal structure of CNN is shown in Figure 4.

E. CNN-LSTM PARALLEL NETWORK

Hybrid CNN-LSTM networks are often configured in series, where CNN is used to extract features from the input data and subsequently, the output of the CNN is fed into the LSTM as an input. Combining CNN and LSTM can make use of their complementary characteristics such as, CNN being used for feature extraction that expresses spatial locality and LSTM being implemented for time series data analysis for temporal feature detection. However, an obvious query to series CNN-LSTM configurations is, to what extent the accuracy of the CNN model affects the training of the LSTM model. To avoid this confusion completely, CNN-LSTM parallel networks can be used where each NN will have its own path without intersecting or affecting each other [31]. The LSTM follows a conventional path and outputs a 1D array. For the CNN path, the convolution layer and pooling layer outputs a 2D array.

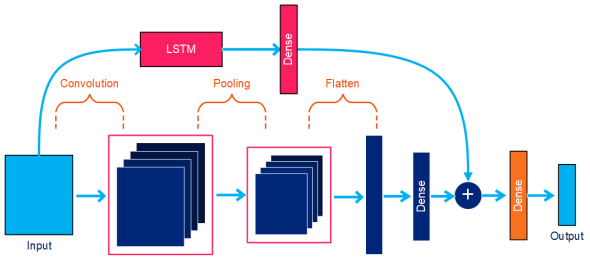

FIGURE 5. Model structure of CNN-LSTM parallel network for temperature forecasting.

However, it has to be ensured that the vector output of the two paths are of the same dimension before they are added. A flatten layer stacks the 2D output of the pooling layer into a 1D array and the dense layer ensures equal number of elements from both the pathways. One significant drawback of this network is the increased computational cost. The model of CNN-LSTM parallel network considered in this study is shown in Figure 5.

F. GRU-LSTM PARALLEL NETWORK

GRU-LSTM hybrid models have previously been proposed for series configuration. To the best of our knowledge, we are the first to implement a GRU-LSTM parallel network for time series prediction. The series configuration was also trained, but the parallel GRU-LSTM yielded better results which is why it is considered for this study. The concept is similar to that of CNN-LSTM; in order to avoid the output of one network adding any bias to the output of another, the series configuration was replaced with a parallel network where each DNN has separate paths for training the data. GRU and LSTM have a similar working mechanism, with GRU being a little faster than LSTM as it has two gates where LSTM has three. Combining the two models have shown promising results.

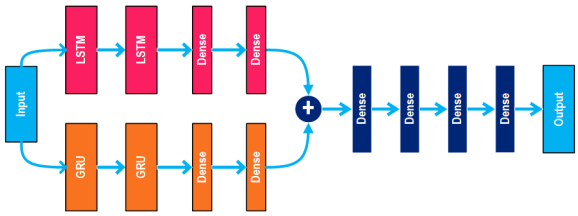

FIGURE 6. Model structure of GRU-LSTM parallel network for temperature forecasting.

Similar to CNN-LSTM parallel network, it has to be ensured that the vector outputs of the two separate NN paths are of the same size before summing them. The combined output enters three dense layers to prepare the data for prediction. The GRU-LSTM parallel network considered in our study is illustrated in Figure 6.

IV. METHODOLOGY

Two different approaches were taken using the six DNNs to perform regression-

1) Considering each hour of the prediction horizon (6h) as an individual regression problem (hour-by-hour prediction).
2) Considering the total prediction horizon (6h) as a single regression problem (full prediction in single run), and

Both approaches are used to predict up to 6h ahead hourly temperature. All six DNN models are evaluated using both approaches. Additionally, the models are trained on data with normalization and compared to the same models trained on data without normalization to see how normalization affects DNNs for univariate time series data. Finally, datasets from four other regions are used to perform a robustness analysis of DNN models. To increase the resolution of the data, a hopping window of hop size equal to 1h is used to divide the data into overlapping blocks of 30h each, for both train and test sets. From each of these blocks, the first 24h is taken as the input sequence and the remaining 6h is taken as the output sequence. This approach is followed for the full prediction

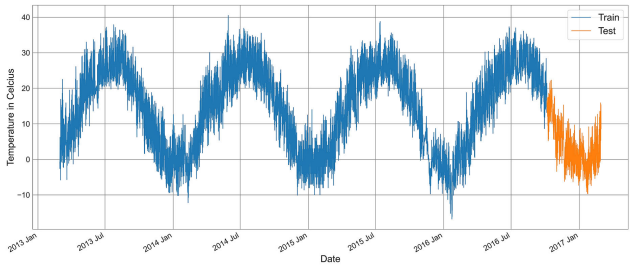

FIGURE 7. The train-test split considered for the Beijing hourly temperature dataset.

in a single run. For the hour-by-hour case, the 6h prediction horizon is considered as six individual regression problems, while the 24h input is kept the same. The models are trained on both normalized data and data without normalization to compare the raw performance of the DNNs, and also observe how normalization affects the performance of the models.

A. DATA COLLECTION

The temperature data is collected from a dataset uploaded by Zhang S. *et al.* titled ''Cautionary Tales on Air-Quality Improvement in Beijing'' [32]. The original data contained various air quality readings from twelve nationally controlled monitoring sites. From the whole dataset, the Aoti Zhongxin area is taken for its relatively low number of missing values. The Aoti Zhongxin is considered in this study to represent overall Beijing temperature because of the low variation in readings from other centers.

The dataset consisted of hourly temperature data from 2013-03-01 00:00:00 to 2017-02-28 23:00:00 giving us a total of 35064 hourly readings. The dataset is at first sorted according to datetime. There were 20 missing temperature values and because of the relatively small size of the missing data, it is filled using the forward fill method instead of other complex imputation methods. Then maintaining the order, first 90% of the data is selected for training from 2013-03-01 00:00:00 to 2016-10-04 18:00:00 and the remaining is taken for testing from 2016-10-04 19:00:00 to 2017-02-28 23:00:00. The train-test split can be visualized from Figure 7.

The dataset for the robustness analysis titled ''Historical Hourly Weather Data 2012-2017'' [33] contains 5 years of high resolution (hourly measurements) temporal data of various weather attributes from January 2012, 12:00:00 to December 2017, 00:00:00, out of which the temperature data is extracted. This data is available for 30 US and Canadian cities. Toronto, Seattle, Dallas and Las Vegas were chosen for the robustness analysis because of their considerably scattered geographical locations so that the temporal data vary as much as possible.

B. MODEL CONSTRUCTION AND HYPERPARAMETER TUNING

Hyperparameter tuning is an important part of NN construction, which is usually done through extensive trial and error. A common practice is to use rule-of-thumb parameters or combinations that have previously performed well for other papers. However, we have carefully chosen all the hyperparameters after manually testing from a wide range of values. A validation run is conducted for each model to decide the hyperparameters for best performance and fitting before training the final models. The train set is split 90-10 for the validation run. The layer-based hyperparameters determined from this run, are provided in the Table 2.

TABLE 2. Model based hyperparameters of the considered DNNs.

SRN, LSTM, GRU: • Activation: ELU • No. of units: 240 • Learning rate: 0.001 • Epoch: 20	**CNN:** • Activation: ELU • No. of units: 240 • Kernel size: 3 • Padding: same • Pooling: max pooling • Pooling kernel size: 4 • Stride: 4 • Learning rate: 0.001 • Epoch: 20
CNN-LSTM: • Activation: ELU • No. of units: 240 (both CNN and LSTM) • Kernel size: 3 • Padding: "same" • Pooling: max pooling • Pooling kernel size: 4 • Stride: 4 • Learning rate: 0.001 • Epoch: 10	**GRU-LSTM:** • Activation: ReLU (1^{st} recurrent and dense layer of both subnetworks) • Activation: Linear for last two layers • Activation: ELU for all the rest. • No. of units: 240 for all • Learning rate: 0.00015 • Regularizer: L2 regularizer (10^{-4}) • Epoch: 20

General parameters such as optimizer, learning rate and the number of epochs are also important to improve the overall performance and speed of the models. Commonly used optimizers include root mean square propagation (RMSprop), stochastic gradient descent (SGD), the adaptive gradient algorithm (AdaGrad), and adaptive moment estimation (Adam). In this paper, after the validation run, the Adam optimizer is chosen which is computationally efficient and showed slightly better results during testing. The batch size of all the models is taken as 64 and the loss functions considered are- mean square error (MSE), cosine similarity (for full time single run) and MSE for hour-by-hour prediction.

V. RESULT ANALYSIS

A. FORECASTING OUTCOMES

The trained models were used to predict hourly temperatures up to 6h ahead. The prediction is carried out for hour-by-hour basis as well as the whole time horizon in a single run. The training and testing period has been mentioned in section IV-A. It is observed that the models trained on unnormalized data perform better than models trained on normalized data, and so only the prediction graphs of models

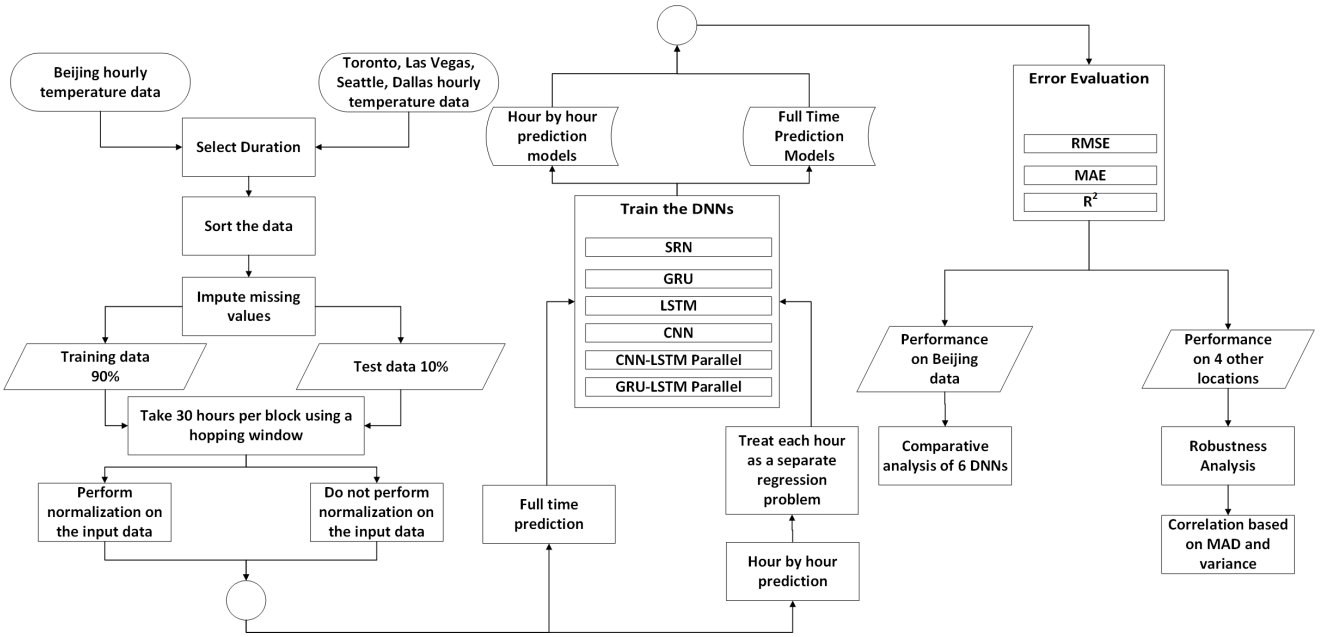

FIGURE 8. Workflow of modeling the considered deep neural networks.

trained on data without normalization are included in the paper (Figure 9 to Figure 20). The error metrics values are tabulated and RMSE is plotted for both with and without normalization.

B. EVALUATION METRICS

The performance of the DNNs are evaluated in terms of three error metrics. The error metrics taken into account are the conventional root mean squared error (RMSE) and mean average error (MAE) and additionally, the coefficient of determination R^2. The mathematical expressions of the above error metrics are given as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(F_t - A_t\right)^2} \tag{14}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\left(F_t - A_t\right)| \tag{15}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(F_t - A_t\right)^2}{\sum_{i=1}^{n}\left(F_t - \bar{A_t}\right)^2} \tag{16}$$

where n is the number of data in forecasted temperature, F_t is the forecasted hourly temperature and A_t is the actual temperature at instant i. For R^2, $\bar{A_t}$ is the mean value of the observations. The R^2 value indicates how good a model fits the dataset. The maximum value of R^2 is 1, where values closer to 1 indicate higher prediction accuracy. RMSE puts more emphasis on higher errors compared to the lower ones. Lower values of RMSE and MAE indicate better performance.

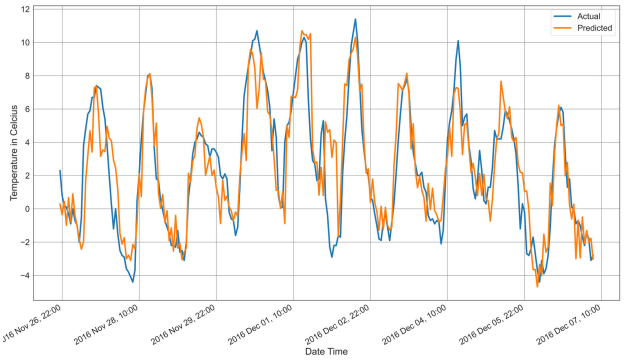

FIGURE 9. Curve of actual temperature and predicted results for hour-by-hour prediction using SRN.

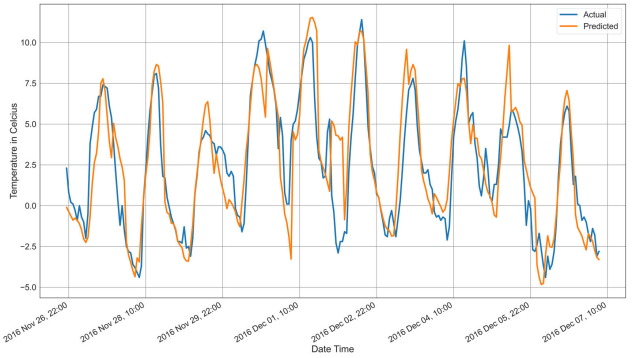

FIGURE 10. Curve of actual temperature and predicted results single run prediction using SRN.

C. PERFORMANCE ASSESSMENT BASED ON EVALUATION METRICS

The performance of the DNNs is assessed mainly based on their RMSE values. The effect of normalization is also

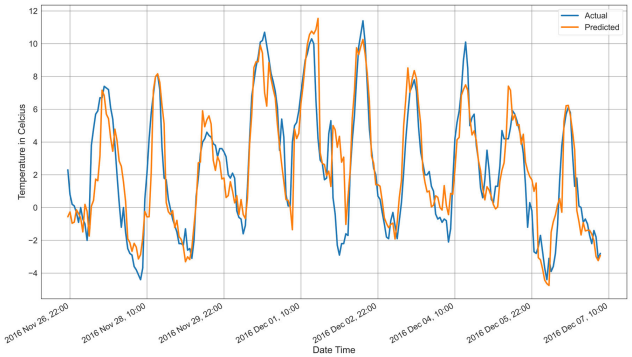

FIGURE 11. Curve of actual temperature and predicted results for hour-by-hour prediction using LSTM.

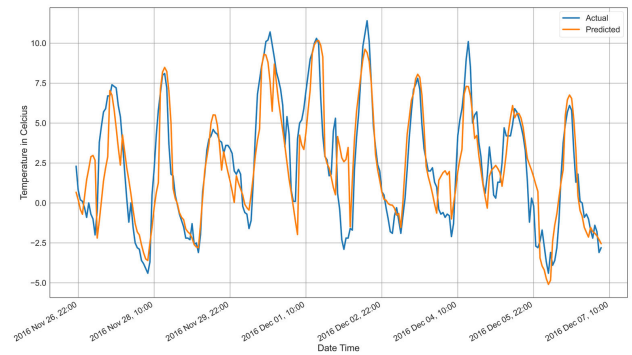

FIGURE 12. Curve of actual temperature and predicted results for single run prediction using LSTM.

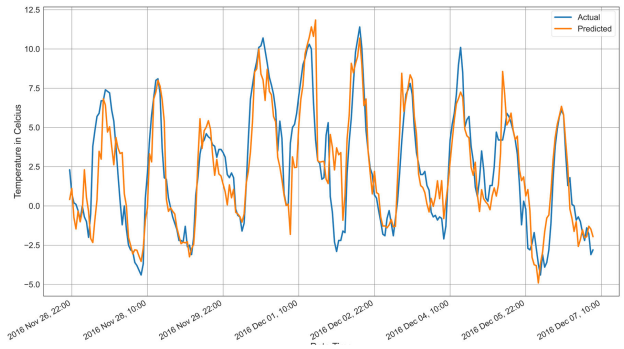

FIGURE 13. Curve of actual temperature and predicted results for hour-by-hour prediction using GRU.

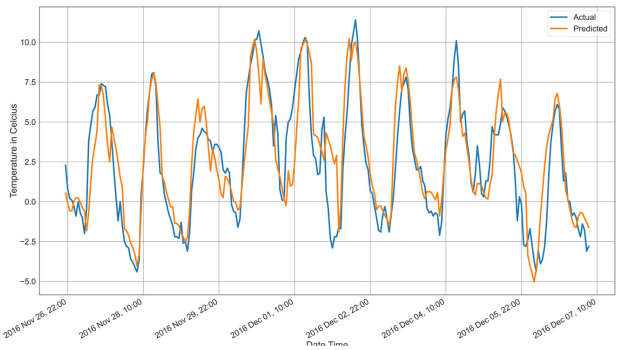

FIGURE 14. Curve of actual temperature and predicted results for single run prediction using GRU.

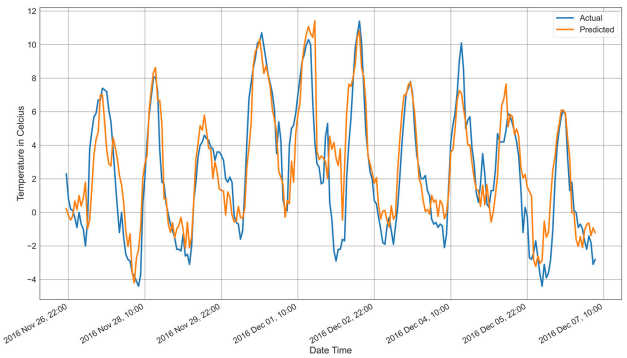

FIGURE 15. Curve of actual temperature and predicted results for hour-by-hour prediction using CNN.

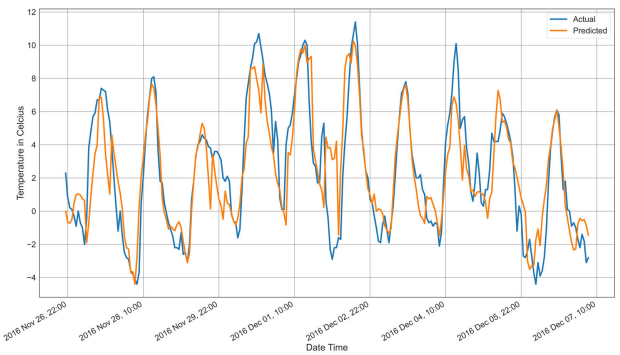

FIGURE 16. Curve of actual temperature and predicted results for single run prediction using CNN.

observed. The error metrics values for models trained on unnormalized data and models trained on normalized are tabulated in Table 3 and Table 4, respectively. The following observations can be extracted from the results:

1) OVERVIEW OF MODEL PERFORMANCE

It can be observed from Figure 21 that SRN had the highest RMSE, followed by GRU and LSTM, which is expected. LSTM is the modified version of SRN, and despite GRU being proposed after LSTM, its main purpose is to reduce computational cost while retaining accuracy as much as possible. Thus, SRN (1.79 for hour-by-hour, 1.88 for full time) and GRU (1.81, 1.79) showed the poorest performance. LSTM (1.77, 1.77) performed slightly better than SRN and GRU. This also reflects the previously mentioned claim that LSTM is more suitable for detecting long-term dependencies rather than high resolution short-term outputs. CNN performed similar to LSTM in both hour-by-hour (1.77) and full-time prediction (1.76) cases.

2) RNNs VS CNN FOR TIME SERIES FORECASTING

In general, RNNs (SRN, LSTM, GRU) are known to work better on text classification whereas CNN is the standard for image classification. According to literature, RNNs work well with sequential data which makes it ideal for predicting values in a sequence (such as time series) while CNN is excellent for feature extractions. However, it can be observed

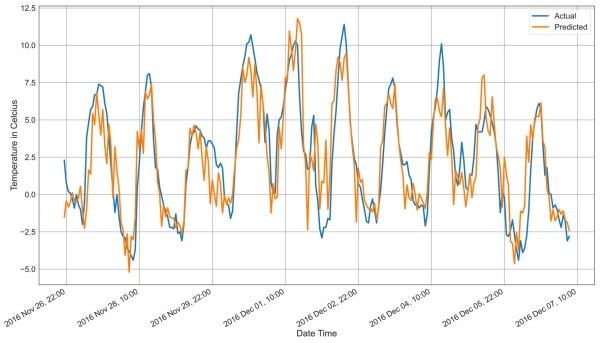

FIGURE 17. Curve of actual temperature and predicted results for hour-by-hour prediction using CNN-LSTM parallel network.

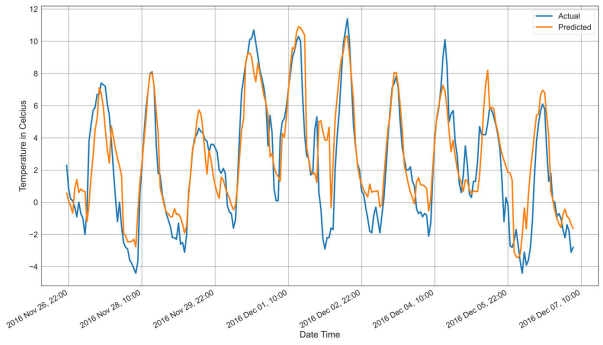

FIGURE 18. Curve of actual temperature and predicted results for single run prediction using CNN-LSTM parallel network.

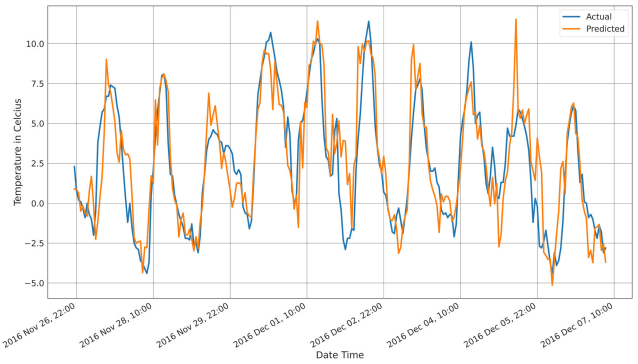

FIGURE 19. Curve of actual temperature and predicted results for hour-by-hour prediction using GRU-LSTM parallel network.

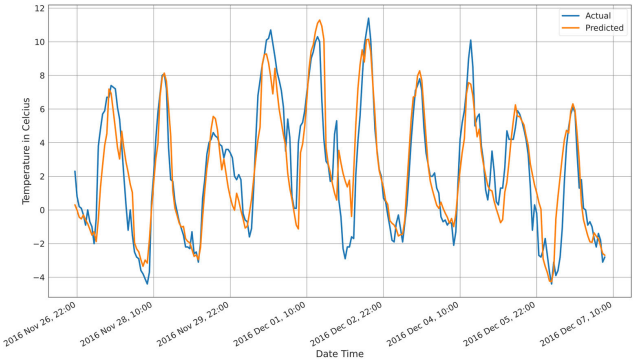

FIGURE 20. Curve of actual temperature and predicted results for single run prediction using GRU-LSTM parallel network.

from Table 3 and Table 4 that they perform similarly on univariate time series predictions. A deciding argument in this regard can be the computation time. CNNs have a huge advantage of being very fast compared to RNNs. In our study, the CNN model ran 5 times faster than LSTM, 4 times faster than GRU and twice as fast as SRN.

3) SINGULAR MODELS VS HYBRID MODELS

An interesting case is observed for the hybrid models CNN-LSTM parallel and GRU-LSTM parallel network. Both models outperformed single models for single-run predictions, yet both showed the worst performance for hour-by-hour predictions. GRU-LSTM exhibited the best RMSE (1.69) for single run, but the second worst hour-by-hour RMSE (1.9) out of all the DNNs. Similarly, CNN-LSTM yielded the second best RMSE (1.74) for single run, and the worst RMSE (2.2) for hour-by-hour prediction. The highly inconsistent performance for the hybrid models, in hour-by-hour prediction can be observed in the form of random spikes in Figure 21 and Figure 22. However, it should be noted that the hybrid models are computationally more expensive than single models.

4) EFFECT OF NORMALIZATION

It is evident from Figure 21 and Figure 22 that the models trained with normalized data performed worse than the models trained without normalization. In Figure 21, only CNN-LSTM parallel network showed random spikes, but in the case of Figure 22, almost every model including SRN, LSTM, GRU-LSTM and CNN-LSTM performed inconsistently. Although normalized data are expected to yield good results on time series forecasting using DNNs, it performed poorly on temperature data.

5) COMPARISON WITH EXISTING WORKS

In Section II, only one paper was found to predict hourly temperature using DNN. They have proposed a convLSTM model which achieved an hourly average RMSE of 2.1° C on a temperature dataset of Germany. Although all the models included in this paper have achieved better RMSE (<2.1° C) for single run prediction, our work cannot be conclusively compared as the works are based on two different datasets.

To summarize this section, the following conclusions were reached:

1) GRU-LSTM parallel network shows superior performance out of all six models (for full time single run prediction, with/without normalization).
2) All DNNs perform better on hourly temperature data without normalization compared to data that is normalized.
3) Full time single run predictions are preferable to hour-by-hour predictions, as hour-by-hour exhibited random spikes. Not to mention the obvious drawback, the hour-by-hour run requires 6 times more computational cost than single runs.

TABLE 3. Evaluation metrics of the considered DNNs trained on Beijing data without normalization.

	Error metrics	SRN		GRU		LSTM		CNN		CNN-LSTM parallel		GRU-LSTM parallel	
Horizon		Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time
1	RMSE	0.9660	0.9912	0.8459	0.8922	0.8646	0.8951	0.8896	0.8837	0.8387	0.8974	0.8724	0.8567
	MAE	0.7011	0.7288	0.5799	0.6228	0.6067	0.6378	0.6363	0.6254	0.5774	0.6526	0.6133	0.6082
	R^2	0.9752	0.9739	0.9810	0.9788	0.9801	0.9787	0.9790	0.9792	0.9813	0.9786	0.9798	0.9805
2	RMSE	1.3266	1.4235	1.3369	1.3378	1.2908	1.3316	1.3128	1.3349	1.8227	1.3434	1.4024	1.2753
	MAE	0.9520	1.0511	0.9757	0.9625	0.9245	0.9647	0.9616	0.9833	1.4132	1.0021	1.0381	0.9262
	R^2	0.9532	0.9462	0.9525	0.9524	0.9557	0.9529	0.9542	0.9526	0.9118	0.9520	0.9477	0.9568
3	RMSE	1.7016	1.7486	1.6455	1.6580	1.6065	1.6590	1.7076	1.6382	3.3979	1.6349	1.6287	1.582
	MAE	1.2706	1.2934	1.2181	1.2290	1.1791	1.2423	1.2890	1.2419	2.6600	1.2460	1.2192	1.1811
	R^2	0.9230	0.9187	0.9280	0.9269	0.9314	0.9268	0.9225	0.9287	0.6933	0.9289	0.9295	0.9335
4	RMSE	1.9214	2.0078	1.9408	1.9319	1.9373	1.9071	1.8858	1.9010	1.9520	1.8721	1.9887	1.8284
	MAE	1.4534	1.5061	1.4471	1.4537	1.4502	1.4473	1.4304	1.4605	1.4905	1.4499	1.5347	1.3854
	R^2	0.9018	0.8928	0.8998	0.9007	0.9002	0.9032	0.9054	0.9039	0.8986	0.9068	0.8948	0.9111
5	RMSE	2.1084	2.2390	2.1816	2.1563	2.1085	2.1199	2.1474	2.0986	2.1661	2.0960	2.3600	2.0391
	MAE	1.6515	1.6950	1.6447	1.6379	1.5908	1.6146	1.6518	1.6029	1.6483	1.6232	1.8170	1.5538
	R^2	0.8817	0.8666	0.8733	0.8762	0.8816	0.8804	0.8773	0.8828	0.8751	0.8830	0.8518	0.8893
6	RMSE	2.3393	2.4731	2.4747	2.3384	2.3867	2.3155	2.2942	2.3024	2.2471	2.2721	2.6279	2.1891
	MAE	1.7992	1.8892	1.8489	1.7814	1.8128	1.7734	1.7699	1.7589	1.7198	1.7576	2.0209	1.6738
	R^2	0.8542	0.8371	0.8369	0.8543	0.8482	0.8572	0.8598	0.8588	0.8655	0.8625	0.816	0.8723
1-6	RMSE	1.7888	1.8809	1.8194	1.7882	1.7743	1.7710	1.7730	1.7590	2.2031	1.7485	1.9066	1.691
	MAE	1.3046	1.3606	1.2857	1.2812	1.2607	1.2800	1.2898	1.2788	1.5849	1.2886	1.3739	1.2214
	R^2	0.9149	0.9059	0.9119	0.9149	0.9162	0.9165	0.9163	0.9177	0.8709	0.9186	0.9033	0.9239

TABLE 4. Evaluation metrics of the considered DNNs trained on Beijing data with normalization.

	Error metrics	SRN		GRU		LSTM		CNN		CNN-LSTM parallel		GRU-LSTM parallel	
Horizon		Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time	Hour by hour	Full time
1	RMSE	1.1729	1.2647	1.2010	1.1745	0.9764	1.0383	1.0270	1.0137	0.8575	1.0360	1.0277	1.4157
	MAE	0.9183	0.9882	0.9583	0.9233	0.7237	0.7805	0.7800	0.7584	0.6060	0.7786	0.7796	1.1454
	R^2	0.9635	0.9575	0.9617	0.9634	0.9747	0.9714	0.9720	0.9727	0.9804	0.9715	0.9719	0.9468
2	RMSE	3.2273	1.5844	1.5061	1.5068	1.5002	1.4920	1.5383	1.3981	1.4562	1.4368	2.4981	1.6342
	MAE	2.9008	1.2020	1.1126	1.1542	1.1081	1.1292	1.1721	1.0531	1.1150	1.0804	2.1541	1.2623
	R^2	0.7235	0.9333	0.9397	0.9397	0.9402	0.9409	0.9371	0.9481	0.9437	0.9452	0.8343	0.9291
3	RMSE	1.7563	1.8372	1.8762	1.7924	2.8586	1.8269	1.7238	1.6804	4.0090	1.7650	1.7563	1.8393
	MAE	1.3518	1.3966	1.4450	1.3903	2.4141	1.4047	1.3105	1.2884	3.2925	1.3511	1.3419	1.4137
	R^2	0.9180	0.9103	0.9064	0.9146	0.7829	0.9113	0.9210	0.9249	0.5730	0.9172	0.9180	0.9101
4	RMSE	2.6829	2.0982	2.0352	2.0275	1.9859	2.1020	1.9690	1.9516	1.8859	1.9920	2.0083	2.044
	MAE	2.0646	1.5990	1.5768	1.5823	1.5458	1.6287	1.5225	1.5183	1.4440	1.5449	1.5517	1.5827
	R^2	0.8086	0.8829	0.8898	0.8907	0.8951	0.8825	0.8969	0.8987	0.9054	0.8944	0.8927	0.8889
5	RMSE	2.2389	2.3358	2.2055	2.2373	2.2016	2.3127	2.1883	2.1499	2.0812	2.1666	2.4151	2.2012
	MAE	1.7428	1.7988	1.7388	1.7546	1.7257	1.8003	1.7036	1.6839	1.5935	1.6970	1.8895	1.7141
	R^2	0.8666	0.8548	0.8705	0.8668	0.8710	0.8576	0.8725	0.8770	0.8847	0.8750	0.8448	0.871
6	RMSE	2.4162	2.5167	2.3224	2.4161	2.3935	2.4815	2.3871	2.3145	2.2151	2.3381	2.4633	2.3428
	MAE	1.8614	1.9646	1.8173	1.9035	1.8755	1.9381	1.8509	1.8118	1.7113	1.8526	1.9266	1.8325
	R^2	0.8445	0.8313	0.8563	0.8445	0.8474	0.8360	0.8482	0.8573	0.8693	0.8544	0.8384	0.8538
1-6	RMSE	2.3425	1.9865	1.8987	1.9068	2.0775	1.9393	1.8599	1.8071	2.2994	1.8431	2.0942	1.9395
	MAE	1.8066	1.4916	1.4415	1.4514	1.5655	1.4469	1.3899	1.3523	1.6270	1.3841	1.6072	1.4918
	R^2	0.8541	0.8950	0.9041	0.9033	0.8852	0.8999	0.9080	0.9131	0.8594	0.9096	0.8833	0.8999

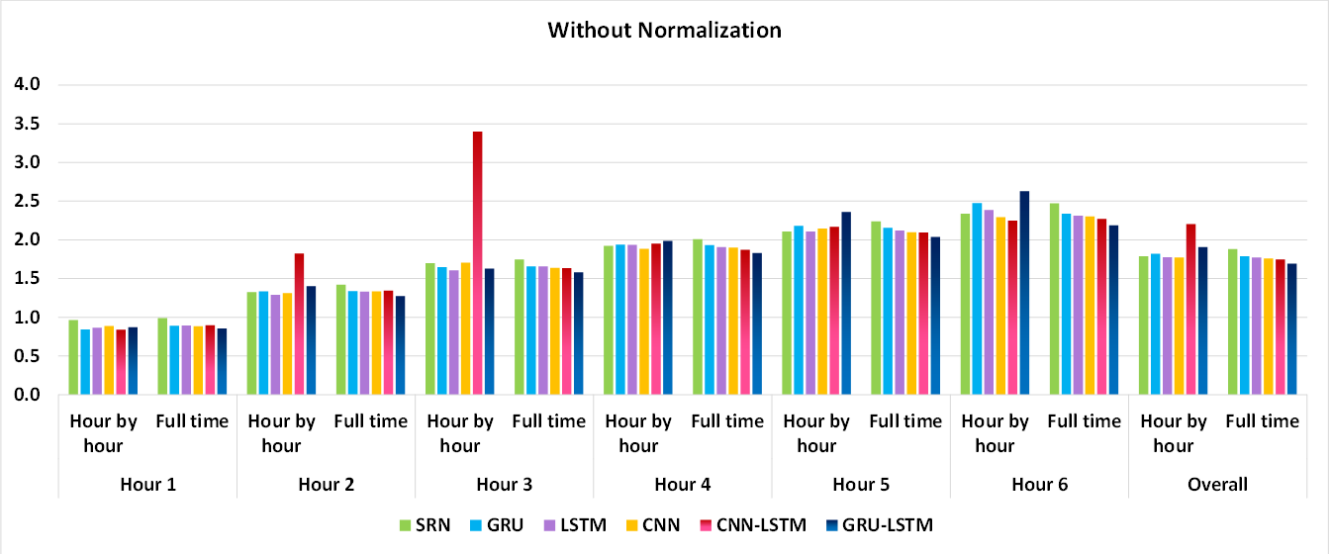

FIGURE 21. RMSE statistics for considered models trained on Beijing dataset without normalization.

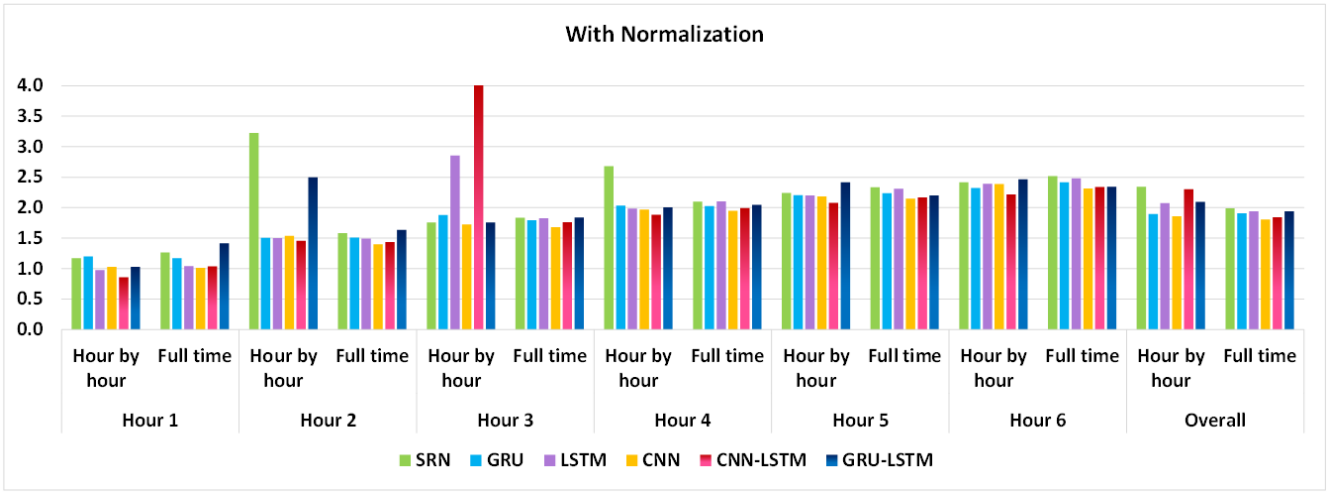

FIGURE 22. RMSE statistics for considered models trained on Beijing dataset with normalization.

4) In terms of computational cost, CNN is much faster than any other model while sustaining good performance.

VI. ROBUSTNESS ANALYSIS

In section V, a conclusion is drawn from the performance of the models by testing them on the same dataset as they trained on. In this section, the robustness of the models are analyzed by testing the models on new datasets from different geographical locations having uncorrelated climatic characteristics. The robustness is a model's ability to generalize trends and output satisfactory performance on different or altered datasets. The previous three error metrics are compared among different DNNs to assess their robustness in each location.

Four cities from different geographical locations were chosen for the robustness analysis, discussed in section IV. The time period considered for the prediction is from 1 March 2013, 00:00:00 to 28 February 2017, 23:00:00 (same as Beijing dataset). The result obtained from the predictions are summarized in Table 5. The models were run with both normalization and without normalization, also hour-by-hour and single-run approaches. Similar to the previous case, models without normalization in a single run yielded better results, so the discussion will be limited to this. To grasp the changes easier, the comparative RMSE of the DNN models is illustrated in Figure 23.

Figure 23 depicts that all the models performed satisfactorily on untrained, unrelated datasets from different locations. The RMSE of all the models did increase, but the increase is comparatively low, indicating a model's robustness and reliability. From Table 5, it can be observed that GRU has achieved the lowest average RMSE (2.0042° C), which indicates that GRU is the most robust DNN.

To draw a correlation between a model's performance and different types of temperature datasets from different regions,

TABLE 5. Robustness analysis of considered DNNs evaluated on four different geographical locations.

Location	Error Metrics	SRN	LSTM	GRU	CNN	CNN-LSTM parallel	GRU-LSTM parallel
Toronto	RMSE	2.1806	2.1660	2.1355	2.0507	1.9987	2.0854
	MAE	1.5392	1.5085	1.4855	1.4514	1.4175	1.4651
	R^2	0.9029	0.9043	0.9069	0.9142	0.9185	0.9112
Las Vegas	RMSE	2.6441	2.5566	2.166	2.4916	2.3266	2.5155
	MAE	1.7332	1.7022	1.6782	1.6401	1.5766	1.6202
	R^2	0.8322	0.8431	0.8245	0.8510	0.8701	0.8482
Seattle	RMSE	1.6244	1.6605	1.5494	1.508	1.4253	1.4701
	MAE	1.2058	1.1785	1.1284	1.0976	1.0551	1.0770
	R^2	0.8863	0.8812	0.8966	0.9020	0.9125	0.9069
Dallas	RMSE	2.4923	2.7241	2.166	2.6643	2.6751	2.6625
	MAE	1.7658	1.8829	1.7593	1.8550	1.8339	1.7944
	R^2	0.8979	0.8780	0.8895	0.8833	0.8823	0.8835
Average	RMSE	2.2353	2.2768	2.0042	2.1786	2.1064	2.1833
	MAE	1.5611	1.5680	1.5128	1.5110	1.4707	1.4891
	R^2	0.8798	0.8766	0.8793	0.8875	0.8958	0.8874

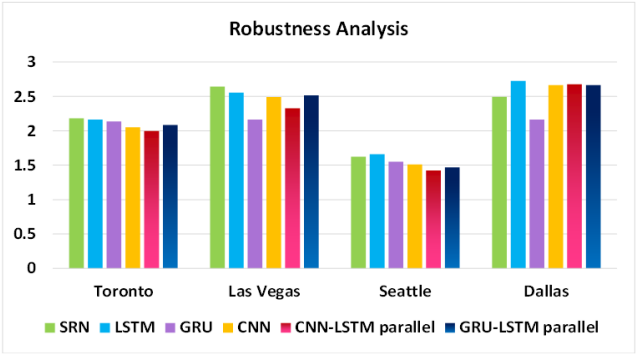

FIGURE 23. RMSE statistics for models evaluated on four different geographical locations.

various parameters were initially considered, such as distribution plot, autocorrelation function (ACF), partial autocorrelation function (PACF), variance, mean absolute deviation (MAD), etc. These parameters did not have any apparent correlation, except the variance and MAD which showed a negative correlation with model performance. Finally, it is observed that the model performance on different datasets is best explained by the product of the variance and MAD.

Table 6 lists the MAD and variance values of the datasets. It indicates that the RMSE value has a positive correlation with the variance and the MAD value of a particular dataset. The MAD value is calculated keeping the Beijing data as a point of reference. Initially, only the variance was considered to draw a correlation. Higher variance in the data caused the models to perform poorly. However, Toronto is an exception where the models performed better despite encountering a very high variance. This can be explained by the second parameter, MAD. Toronto has the lowest MAD value among

TABLE 6. Variance and MAD of the considered geographical locations.

	Variance	MAD	MAD*Variance
Beijing	38.6595	-	-
Toronto	50.2773	4.7294	237.7814
Las Vegas	42.2605	9.1500	386.6835
Seattle	23.3225	5.1871	120.9761
Dallas	61.8568	11.5431	714.0192

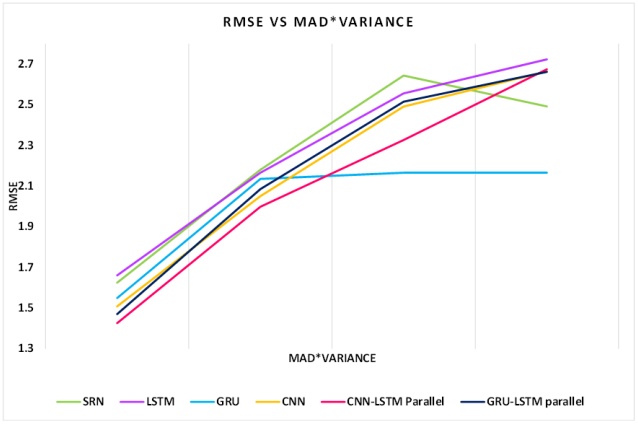

FIGURE 24. Correlation between model performance and MAD*variance.

the four regions. So, the best fit for correlating the model performance with the type of regional temperature dataset are considered as the product of the variance and MAD values. The RMSE of all six models are plotted against the MAD∗variance in Figure 24.

Figure 24 illustrates that LSTM, GRU, CNN, CNN-LSTM, GRU-LSTM all perform worse as the MAD*variance increases indicated by the upward trend of the RMSE values

(except SRN which produced an outlier). Another important point to note is that, for Seattle, all the models have yielded a lower RMSE value compared to Beijing, as it has a lower variance. This implies that the models are able to achieve a degree of generality. On the other hand, the specificity of the models can be understood from the positive correlation of RMSE to the MAD value. This opens the scope of using transfer learning for datasets that have little correlation to the dataset models were trained on.

VII. CONCLUSION

This study has carried out a comparative analysis on six DNN models to observe which performs the best for high-resolution hourly temperature forecasting on Beijing temperature data. The study has also presented an in-depth robustness analysis to see the change in performance parameters of these DNNs when tested on a geographically diverse dataset. The comparative analysis has revealed GRU-LSTM parallel network to provide the best performance when tested on the Beijing data at 1.691° C RMSE. CNN on the other hand performs slightly worse at 1.759 ° C RMSE ranking 3^{rd} in terms of accuracy but has by far the best computational time. The study has also found out that single-run models are better and more consistent for prediction instead of single-point regression models. The comparative analysis further revealed that the models perform poorly on normalized temperature data which is unusual as neural network models generally tend to perform better on normalized data. In short, this study aimed to act as a benchmark for high-resolution temperature forecasting with only historical temperature data using neural nets that yield sufficient accuracy and are computationally inexpensive.

From the robustness analysis, the study was able to map a correlation between model performance and the product of MAD and variance of the dataset. It was further found that the GRU-based model was able to generalize the most over various geographical locations although it performed poorly on Beijing data. This was explained by the high variability of temperature data across the globe. To perform well on temperature data of a particular location, the models had to trade off robustness for a certain level of specificity. This has indicated a future scope of work where transfer learning can be adopted so that models trained on one dataset can perform well on new data with little correlation with the previous dataset. Moreover, this study can be incorporated with research on embedded systems equipped with artificial intelligence processing capabilities to be used in the future to implement portable, compact devices for on-spot temperature forecasting.

REFERENCES

[1] S. S. Sharif and J. H. Taylor, "Real-time load forecasting by artificial neural networks," in *Proc. Power Eng. Soc. Summer Meeting*, Jul. 2000, pp. 496–501.

[2] J. Verhelst, G. Van Ham, D. Saelens, and L. Helsen, "Model selection for continuous commissioning of HVAC-systems in office buildings: A review," *Renew. Sustain. Energy Rev.*, vol. 76, pp. 673–686, Sep. 2017.

[3] J. Zhao and X. Liu, "A hybrid method of dynamic cooling and heating load forecasting for office buildings based on artificial intelligence and regression analysis," *Energy Buildings*, vol. 174, pp. 293–308, Sep. 2018.

[4] D. H. C. Chow and G. J. Levermore, "New algorithm for generating hourly temperature values using daily maximum, minimum and average values from climate models," *Building Services Eng. Res. Technol.*, vol. 28, no. 3, pp. 237–248, Aug. 2007.

[5] J. Shao and P. J. Lister, "An automated nowcasting model of road surface temperature and state for winter road maintenance," *J. Appl. Meteorol.*, vol. 35, no. 8, pp. 1352–1361, Aug. 1996.

[6] J. Bogren and T. Gustavsson, "Site specific road surface temperature forecast improvements by use of radiation measurements," in *Proc. 11th SIRWEC Conf.*, 2002, pp. 1–5.

[7] K. S. Kim, S. E. Taylor, M. L. Gleason, and K. J. Koehler, "Model to enhance site-specific estimation of leaf wetness duration," *Plant Disease*, vol. 86, no. 2, pp. 179–185, Feb. 2002.

[8] J. Cheng, Z. Xu, H. Bambrick, H. Su, S. Tong, and W. Hu, "The mortality burden of hourly temperature variability in five capital cities, Australia: Time-series and meta-regression analysis," *Environ. Int.*, vol. 109, pp. 10–19, Dec. 2017.

[9] G. Papacharalampous, H. Tyralis, and D. Koutsoyiannis, "Predictability of monthly temperature and precipitation using automatic time series forecasting methods," *Acta Geophys.*, vol. 66, no. 4, pp. 807–831, Aug. 2018.

[10] J. Kihoro, R. Otieno, and C. Wafula, "Seasonal time series forecasting: A comparative study of ARIMA and ANN models," Meru Univ., Nairobi, Kenya, Tech. Rep., 2004.

[11] A. Khotanzad, R. Afkhami-Rohani, T.-L. Lu, A. Abaye, M. Davis, and D. J. Maratukulam, "ANNSTLF—A neural-network-based electric load forecasting system," *IEEE Trans. Neural Netw.*, vol. 8, no. 4, pp. 835–846, Jul. 1997.

[12] H. Shah, R. Ghazali, and N. M. Nawi, "Using artificial bee colony algorithm for MLP training on earthquake time series data prediction," 2011, *arXiv:1112.4628*.

[13] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Combining neural networks and ARIMA models for hourly temperature forecast," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. IJCNN Neural Comput., New Challenges Perspect. New Millennium*, Jul. 2000, pp. 414–419.

[14] K. Methaprayoon, W. J. Lee, S. Rasmiddatta, J. R. Liao, and R. J. Ross, "Multistage artificial neural network short-term load forecasting engine with front-end weather forecast," *IEEE Trans. Ind. Appl.*, vol. 43, no. 6, pp. 1410–1416, Nov. 2007.

[15] V. Vamitha, M. Jeyanthi, S. Rajaram, and T. Revathi, "Temperature prediction using fuzzy time series and multivariate Markov chain," *Int. J. Fuzzy Math. Syst.*, vol. 2, no. 3, pp. 217–230, 2012.

[16] T. T. K. Tran, T. Lee, J.-Y. Shin, J.-S. Kim, and M. Kamruzzaman, "Deep learning-based maximum temperature forecasting assisted with meta-learning for hyperparameter optimization," *Atmosphere*, vol. 11, no. 5, p. 487, May 2020.

[17] Z. Zhang and Y. Dong, "Temperature forecasting via convolutional recurrent neural networks based on time-series data," *Complexity*, vol. 2020, pp. 1–8, Mar. 2020.

[18] D. Kreuzer, M. Munz, and S. Schlüter, "Short-term temperature forecasts using a convolutional neural network—An application to different weather stations in Germany," *Mach. Learn. With Appl.*, vol. 2, Dec. 2020, Art. no. 100007.

[19] S. Lee, Y.-S. Lee, and Y. Son, "Forecasting daily temperatures with different time interval data using deep neural networks," *Appl. Sci.*, vol. 10, no. 5, p. 1609, Feb. 2020.

[20] T. Toharudin, R. S. Pontoh, R. E. Caraka, S. Zahroh, Y. Lee, and R. C. Chen, "Employing long short-term memory and Facebook prophet model in air temperature forecasting," *Commun. Statist. Simul. Comput.*, pp. 1–24, Jan. 2021.

[21] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, *arXiv:1506.00019*.

[22] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," Université de Montréal, Montréal, QC, Canada, Tech. Rep., 2001.

[23] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[24] R. K. Agrawal, F. Muchahary, and M. M. Tripathi, "Long term load forecasting with hourly predictions based on long-short-term-memory networks," in *Proc. IEEE Texas Power Energy Conf. (TPEC)*, Feb. 2018, pp. 1–6.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[26] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2326–2335.
[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014, *arXiv:1409.3215*.
[28] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*.
[29] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3075–3084.
[30] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2021, Art. no. 107398.
[31] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, "On short-term load forecasting using machine learning techniques and a novel parallel deep LSTM-CNN approach," *IEEE Access*, vol. 9, pp. 31191–31212, 2021.
[32] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in Beijing," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 473, no. 2205, Sep. 2017, Art. no. 20170457.
[33] R. Tatman. (Nov. 2017). *R vs. Python: The Kitchen Gadget Test, Version 1*. Accessed: Dec. 20, 2017. [Online]. Available: https://www.kaggle.com/rtatman/r-vs-python-the-kitchen-gadget-test

EHTASHAMUL HAQUE was born in Dhaka, Bangladesh. He is currently pursuing the B.Sc. degree in electrical and electronic engineering with the Islamic University of Technology, Gazipur, Bangladesh. His main research interests include smart grid and machine learning.

SANZANA TABASSUM (Student Member, IEEE) is pursuing the B.Sc. degree in electrical and electronic engineering with the Islamic University of Technology, Gazipur, Bangladesh. Her main research interests include renewable energy, smart grid, and machine learning.

EKLAS HOSSAIN (Senior Member, IEEE) received the B.S. degree in electrical and electronic engineering from the Khulna University of Engineering and Technology, Bangladesh, in 2006, the M.S. degree in mechatronics and robotics engineering from the International Islamic University of Malaysia, Malaysia, in 2010, and the Ph.D. degree from the College of Engineering and Applied Science, University of Wisconsin–Milwaukee (UWM). He was working in the area of distributed power systems and renewable energy integration for last ten years and has published a number of research papers and posters in this field. Since 2015, he has been involved with several research projects on renewable energy and grid tied microgrid system at the Department of Electrical Engineering and Renewable Energy, Oregon Tech, as an Assistant Professor. He is currently working as an Associate Researcher at the Oregon Renewable Energy Center (OREC). His research interests include modeling, analysis, design, and control of power electronic devices; energy storage systems; renewable energy sources; integration of distributed generation systems; microgrid and smart grid applications; robotics, and advanced control systems. He is a Senior Member of the Association of Energy Engineers (AEE). He is a Registered Professional Engineer (PE) in OR, USA. He is also a Certified Energy Manager (CEM) and a Renewable Energy Professional (REP). He is the Winner of the Rising Faculty Scholar Award from the Oregon Institute of Technology for his outstanding contribution in teaching, in 2019. He is serving as an Associate Editor for IEEE Access.