Numerical Weather Forecasting Using Convolutional-LSTM With Attention and Context Matcher Mechanisms

Selim Furkan Tekin, Arda Fazla, and Suleyman Serdar Kozat, Senior Member, IEEE

Abstract—Numerical weather forecasting with high-resolution physical models requires extensive computational resources on supercomputers, often making it impractical for real-life applications. Alternatively, deep learning methods can provide results within minutes of receiving data. Although baseline deep learning models can make accurate short-term predictions, their performance deteriorates rapidly as the output sequence length increases. However, many real-life scenarios require long-term prediction of certain weather features to mitigate and take advantage of the effects of high-impact weather events. In response, we introduce the Weather Model, which provides rapid and accurate long-term spatial predictions for high-resolution spatio-temporal weather data. We integrate a stacked convolutional long-short term memory (ConvLSTM) network as our building block, given its accuracy in capturing spatial data patterns through convolution operations. Furthermore, we additionally incorporate attention and context matcher mechanisms. The attention mechanism allows the effective usage of the side-information vector by selectively focusing on different parts of the input sequence at each time step. Concurrently, the context matcher mechanism enhances the network’s ability to preserve long-term dependencies. Our Weather Model achieves significant performance improvements compared to baseline deep learning models, including ConvLSTM, TrajGRU, and U-Net. Our experimental evaluation involves high-scale, real-world benchmark numerical weather datasets, namely the ERA5 hourly dataset on pressure levels and WeatherBench. Our results demonstrate substantial improvements in identifying spatial and temporal correlations, with attention matrices focusing on distinct parts of the input series to model atmospheric circulations. We also compare our model with high-resolution physical models using benchmark metrics to confirm our Weather Model’s accuracy and interpretability.

Index Terms—Attention, convolutional long-short term memory (ConvLSTM), convolutional neural network (CNN), numerical weather forecasting.

I. INTRODUCTION

Weather is a vital scientific research area as it is the cornerstone of every real-life system such as agriculture, energy production, tourism, transport, navigation, and climate [1], [2]. Accurate forecasts prevent life and economic losses and support emergency management by diminishing the effects of high-impact weather events and creating substantial financial revenue [3]. Foundations of modern-day weather forecasting were introduced in the early 20th century when scientists solved a system of nonlinear differential equations by hand [4]. Today, supercomputers solve these equations for millions of points per time step to accurately forecast up to months ahead [5]. Meteorologists call this problem numerical weather prediction (NWP), where the objective is to predict a numerical value of a weather feature such as temperature or wind speed for a location. However, NWP models are getting more complex, and their demand for high computation power increases continuously [5]. Obtaining results from these models can take hours, days, or sometimes weeks, limiting their ability to provide actionable predictions in a short time.

We focus on NWP for multiple locations, a spatio-temporal time series problem. A spatio-temporal time series contains multiple time series belonging to different locations with temporal and spatial covariances, such as the temperature at multiple weather stations in a city. Hence, NWP is a spatio-temporal prediction problem on a grid. In this grid structure, compared to commonly used physical models for NWP, deep learning models can provide results within minutes of receiving data, exploit the big data aggregated in years, and make accurate predictions with significantly less computational resources [6]. The success of spatio-temporal time series forecasting with deep learning has emerged with the introduction of convolutional long-short term memory (ConvLSTM) cells in [7], a modified version of long-short term memory (LSTM) cells of [8] that is targeted to capture spatial patterns or wind speed for a location. However, NWP models are

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
human intervention. Therefore, more data sources and novel structures are needed to model such chaotic behaviors [5].

Here, we call our target series the endogenous variable and the different data sources entering the forecasting model as the side-information or exogenous variables. The side-information can be concatenated with the input series and directly given to the model. However, the side-information integration in previous works shows more effective ways [14], [15], [16]. Qin et al. [17] demonstrated that implementing fully connected neural networks (FC-NNs) before the memory units allows the model to specifically focus on the patterns of the exogenous series that are correlated with the endogenous series. This process is known in the literature as attention and was introduced in [18] and [19]. Even though FC-NNs are effective, they cannot capture spatial information as well as convolutional neural networks (CNNs). Thus, Zhang et al. [20] introduced a convolutional attention mechanism for gesture recognition and indicated the need for spatial attention in ConvLSTM structures. Moreover, Han et al. [10] and Agrawal et al. [21] demonstrated the high performance of the U-net model in weather forecasting, which contains multiple layers of CNNs with skip connections. Similar to [7], [11], and [22], U-net follows the idea of encoding input spatio-temporal time series to a small abstract representation and decoding this representation to generate an output sequence. The advantage of an encoder–decoder structure is that the length of the input and output sequences may differ, allowing forecasts of varying lengths. Recently, Rasp et al. [23] released a benchmark dataset from the ERA5 archive, introduced evaluation metrics to make inter-comparison between studies, and provided baseline scores for multiple baseline deep learning and physical forecasting models. These baseline deep learning models make accurate short-term predictions. However, their performance deteriorates rapidly as the length of the output sequence increases.

To this end, we introduce a novel deep learning architecture for NWP by combining multiple spatio-temporal data sources with an attention mechanism and using ConvLSTM cells as building blocks to capture spatial and temporal correlations. We illustrate our complete model architecture in Fig. 1(a). Our model selects relevant cells in multiple spatio-temporal time series to make long-term spatial predictions by preserving the long-term dependences using attention and context matcher mechanisms. Thus, we introduce an encoder–decoder architecture specific to NWP. First, our attention-based encoder block consisting of stacked ConvLSTM units encodes the input sequence to hidden states. Then, our context matcher mechanism aligns the decoder’s hidden states by summing the encoder’s hidden states across all time steps. This process addresses long-term dependences by extending the length of the gradient flows. Each ConvLSTM unit in the decoder block decodes the passed information from the hidden layers of the encoder. The output of the decoder is then passed through the output convolutions to produce predictions. Altogether, our model architecture is adept at focusing on different parts of the spatio-temporal time series and successfully capturing long-term dependences. We call our architecture the Weather Model. We demonstrate significant performance gains over benchmark deep learning models through extensive experiments on two real-life datasets from the ERA5 archive. Furthermore, we openly share our source code to facilitate further research and comparison.

A. Prior Art and Comparisons

Applying deep learning methods to weather forecasting is an extensively studied area. Liu et al. [24] leveraged a massive volume of weather datasets to train an autoencoder with multilayered neural networks (NNs) to forecast hourly weather data. Hossain et al. [25] improved this approach by introducing stacked autoencoders to temperature forecasting. Later, Lee et al. [26] implemented a simple NN to forecast rainfall using historical rainfall data and lagged climate indices from various stations. These works demonstrated the high performance of simple deep learning architectures and their potential with high volume of data. Moreover, Kanigor and Salman [27] and Zaytar and El Amranis [28] applied recurrent NNs (RNNs) and LSTMs to weather forecasting and Poornima and Pushpalatha [29] focused on the vanishing gradient problem by implementing an intensified LSTM architecture for rainfall forecasting. Wang et al. [30] introduced deep uncertainty quantification using an RNN-based architecture and obtained a single-value forecast. Recently, Hewage et al. [31] used temporal convolutional network (TCN) for multiple-input single-output weather predictions and demonstrated superior results with respect to a physical NWP model, namely the weather research and forecasting (WRF) model [32]. However, this approach is costly, does not comprise multiple features and spatial covariances, and only provides high performance for multiple-input single-output models.

As a remedy, Shi et al. [7] introduced ConvLSTM and convolutional gated recurrent units (ConvGRUs) to the forecasting network, which is an encoder–decoder structure consisting of stacked ConvLSTMs or ConvGRUs. These models capture the spatial-temporal covariances and demonstrate high performance in rainfall intensity prediction by using radar echo sequences. Woo and Wong [33] also showed that ConvLSTM is more accurate than the optical flow model and better at capturing spatio-temporal correlations than the fully connected LSTM. Later, Shi et al. [11] introduced the trajectory gated recurrent unit (Traj-GRU) to learn location invariant movements between input radar images, which achieved slightly higher performance than ConvGRU. Although ConvLSTM and TrajGRU offer high performance, they are limited by the encoder–decoder structure, which restricts the output production to the dimensionality of the given input. As a remedy, Tran and Song [22] improved the encoder–decoder architecture by integrating downsampling and upsampling layers between TrajGRU units and eliminated the issue of single-dimensional input through the addition of an output convolution layer. Furthermore, Heye et al. [34] employed a recursive decoder prediction method in the encoder–decoder architecture to enhance the accuracy of long-term forecasts. To further show the learning capacity of ConvLSTMs, Jing et al. [35] implemented a generative adversarial NN (GAN) with ConvLSTM units to extrapolate echoes to a higher resolution.
Fig. 1. Graph of our Weather Model is demonstrated in order to further clarify the recursive output generation process. (a) Details of the encoder, decoder, attention, and context matcher architectures. (b) Inside of a ConvLSTM cell. (c) Unrolled view of the decoder architecture.

using adversarial loss. Moreover, U-Net demonstrated high performance in precipitation nowcasting [10], [21], achieving results comparable to those of ConvLSTMs.

The usage of ConvLSTM-related works is widely evident in recent weather forecasting literature due to their accuracy in capturing spatial covariances [36], [37], [38]. Kim et al. [36] proposed a deep learning model that combines a denoising autoencoder and ConvLSTM to predict the ocean weather worldwide. The proposed Weather Model was highly applicable for predicting ocean weather for up to a week. Hewage et al. [39] acknowledged the need for a deep learning alternative to exhaustive physical models and introduced a lightweight and novel weather forecasting system that utilizes a TCN and long short-term memory (LSTM) network. Moreover, Kreuzer et al. [37] employed a ConvLSTM network for short-term temperature forecasting at various weather stations in Germany. Similarly, Castro et al. [40] developed a deep learning architecture that only uses multiple convolutional layers to learn spatial and temporal dependences for rainfall and air temperature data in South America. Furthermore, Diaconu et al. [38] directly applied the ConvLSTM architecture with four stacked layers for weather forecasting and conducted a thorough comparison with several state-of-the-art baseline models using the EarthNet2021 dataset. Generally, ConvLSTM-related works in the weather forecasting literature either directly apply the ConvLSTM architecture to new weather datasets or combine stacked ConvLSTM models with other deep learning models, like autoencoders or NN variants. Our approach, however, involves using multiple stacked ConvLSTM units as the building blocks of an encoder-decoder architecture while significantly enhancing the model performance with the integration of attention and context matcher mechanisms, as we demonstrate in our simulations. Therefore, our Weather Model distinctly stands out from the existing ConvLSTM-related works in the weather forecasting literature.

Attention mechanisms in the spatio-temporal domain are studied in various tasks. For human action recognition, Song et al. [41] developed temporal and spatial attention mechanisms with fully connected NNs and LSTMs. Similarly, Qin et al. [17] implemented a dual attention mechanism in time series forecasting. Liang et al. [42] implemented a multilevel attention mechanism fusing the side-information with multiple geo-sensory time series data to forecast air quality. For traffic prediction, Yao et al. [43] implemented an attention mechanism on the outputs of LSTM, focusing on different time periods from the past. Similarly, Ma et al. [9]
combined a specific local attention memory mechanism with LSTM to capture spatio-temporal dependences in precipitation nowcasting. Although these works successfully implement an attention mechanism, the input is bound to be single or multiple time series rather than 3-D tensors. As a solution, Zhang et al. [20] implemented an attention mechanism that took 3-D tensors as input. However, the performance increase was limited. Recently, Abdellauoi and Mehrkanoon [44] conducted a comparison study between an LSTM built on top of a CNN block (Conv + LSTM) and a ConvLSTM with temporally unrolled and temporally concatenated architectures, where the ConvLSTM outperformed the Conv + LSTM configuration.

Finally, to facilitate a comprehensive evaluation and comparison of recently emerging deep learning models, Rasp et al. [23] introduced a benchmark dataset called ERA5, which is based on predicting global weather patterns in advance. In addition, the dataset involves baseline scores from simple linear regression techniques, deep learning models, and purely physical forecasting models.

B. Contributions

Our main contributions are as follows.

1) We introduce a novel attention-based recurrent network architecture for spatio-temporal time series called the Weather Model, which incorporates convolution operations on attention layers and utilizes ConvLSTMs as memory units. Our model focuses on the relevant cells in exogenous series to successfully learn spatial correlations and make accurate predictions.

2) To the best of our knowledge, for the first time in the literature, we construct a specific attention-based ConvLSTM network for spatio-temporal weather forecasting. Our model architecture allows us to input multiple exogenous series and produce refined predictions with high interpretability for each series. In addition, our model is easily applicable to various scenarios, as we can make hourly, daily, or any temporal resolution forecasts.

3) We compare our Weather Model with the state-of-the-art deep learning models in the literature, as well as physical models, in order to illustrate the superiority of our approach. We showcase our results through extensive experiments over well-known real-life datasets.

4) We publicly share the implementation of our method, with open-access datasets and an accompanying open-access reproducible notebook, for model design, comparisons, and experimental reproducibility.1

II. PROBLEM DESCRIPTION

In this article, all vectors are real column vectors and are presented in boldface lowercase letters. We represent matrices as bold upper letters, such as \( X, Y, \) and \( W \) and denote the multidimensional tensors by uppercase letters with calligraphic fonts such as \( \mathcal{X}, \mathcal{Y}, \) and \( \mathcal{W} \). The time index is given as subscript, and superscript refers to other signals. For example, \( X_t \) shows \( t \)th input signal at the time step \( t \), which denotes a matrix. Similarly, \( X_t \) shows a multidimensional tensor, representing all weather features at time step \( t \), and \( X^l \) also shows a multidimensional tensor, representing the values of the \( l \)th weather feature for all time steps. In addition, we show the entries of a matrix with lowercase letters with subscripts referring to its position: for example, \( a_{i,j} \) shows the entry of matrix \( A \) at the \( i \)th row and \( j \)th column. Similarly, \( x_{i,1,t} \) denotes the scalar value of the \( l \)th weather feature at time step \( t \) in the first spatial dimensions.

We aim to forecast the weather value of multiple locations using multiple features of historical numeric weather data. The input is the weather features with the grid representation creating a set of spatio-temporal time series. The weather values of a feature \( l \) at a time step \( t \) can be illustrated as a matrix, \( X^l_t = \{ x_{1,1,t}^l, \ldots, x_{i,j,t}^l, \ldots, x_{M,N,t}^l \} \in \mathbb{R}^{M \times N} \), where \( M \) and \( N \) define the spatial dimensions and \( x_{i,j,t}^l \) represents the cell value in the grid at the \( i \)th row and \( j \)th column.

Following this definition, for a given time window \( T \), \( X^l = \{ X^l_1, \ldots, X^l_T \} \in \mathbb{R}^{M \times N \times T} \) represents the spatio-temporal time series for the weather feature \( l \). Similarly, for the total number of \( L \) features, \( X = \{ X^1, \ldots, X^L \} \in \mathbb{R}^{M \times N \times L} \) represents all weather features at time step \( t \). Altogether, we represent the set of spatio-temporal time series as \( X \in \mathbb{R}^{M \times N \times T} \) capturing all of the grid information of \( L \) features until time \( T \). Next, we define \( Y_t = \{ Y^1_t, \ldots, Y^L_t \} \in \mathbb{R}^{M \times N} \) as our endogenous variable, and we represent the target series as \( \mathcal{Y} = \{ Y_1, \ldots, Y_T \} \in \mathbb{R}^{M \times N \times T} \).

Here, our goal is to predict the next sequence of the target series given the previous samples of the input sequences. Mathematically, our objective is to learn the mapping

\[
\{ X_1, \ldots, X_T \} \xrightarrow{\mathcal{F}} \{ Y_{T+1}, \ldots, Y_{T+n} \}
\]

where \( T_{in} \) and \( T_{out} \) are the input and output sequence lengths, respectively. Note that \( T_{out} \) can be any number of time steps, e.g., we can get one-step ahead prediction, \( T_{out} = 1 \), or multistep ahead prediction, \( T_{out} = 7 \), etc. We suffer the mean squared error (MSE) given as

\[
\mathcal{L} = \frac{1}{T_{out}MN} \sum_{t=T_{in}+1}^{T_{in}+T_{out}} \sum_{i=1}^{M} \sum_{j=1}^{N} (y_{i,j,t} - \hat{y}_{i,j,t})^2
\]

where \( \hat{y}_{i,j,t} \) and \( y_{i,j,t} \) represent the predicted and ground truth cell values, respectively. We aim to minimize the loss (1).

Although we use the MSE as the loss measure since it is widely used in the community, any other differentiable loss function can be readily used instead of the MSE in our algorithm, as shown in this article.

III. WEATHER MODEL

In this section, we explain our Weather Model architecture, referring to every step and formulation as illustrated in Fig. 1. First, we provide a comprehensive explanation of our model architecture. Then, we briefly explain each key component present in the architecture.

Fig. 1(a) presents the overall representation of our Weather Model. First, the input series are fed through the attention mechanism to create a weighted input series. This is satisfied

1https://github.com/sftekin/spatio-temporal-weather-forecasting
by multiplying all weather features at each time step with attention matrices. The attention matrices are obtained from the previous hidden state of the encoder’s first layer. Overall, the attention network works as an initial layer that helps the encoder to effectively capture the necessary information in the input sequence. These weighted input series then pass through the encoder layer, which consists of a stacked ConvLSTM block. Here, a separate hidden state is created for each ConvLSTM cell. The ConvLSTM architecture is employed to capture the spatial-temporal covariances in the data. Thus, the attention network, followed by the ConvLSTM encoder unit, work together to gather meaningful information from the input series. Next, the context matcher mechanism is employed to align these hidden states by aggregating the hidden states from the encoder across all time steps so far, which helps to maintain long-term dependences by satisfying the gradient flow from the first data sample to the last. Finally, the output of the context matcher is passed through the decoder, which is also a stacked ConvLSTM block. Hence, the encoder and decoder architectures consist of end-to-end stacked ConvLSTM modules with attention and context matcher mechanisms to further enhance their performances. We also provide the inside of the attention and context matcher mechanisms in Fig. 1, which are thoroughly explained in Section III-B.

In Fig. 1(b), we illustrate the inner structure of a single ConvLSTM cell, where the mathematical foundations are explained in Section III-A. In Fig. 1(c), we provide the unraveled view of our encoder and decoder architectures to better understand how our Weather Model produces its prediction outputs. Here, we illustrate how our model recursively produces its outputs for the following time steps. At each time step, the attention mechanism takes the previous hidden state of the encoder’s first layer as the input to calculate the attention weights. This process is recursively continued for the following time steps. Similarly, the decoder takes the target sequence from the last time step as input, along with the output of the context matcher mechanism. After the first prediction, the prediction of the Weather Model for the last time step is provided to the decoder in place of the target sequence to satisfy recursive prediction for future time steps.

Next, we briefly explain each key component present in our Weather Model.

A. Capturing Spatio-Temporal Relations With ConvLSTMs

One of the critical temporal models in deep learning is the RNN [45]. RNNs store the temporal information of the input sequence within their hidden state vectors, i.e., the memory. The ability to carry information through time in their memory makes RNNs successful in time series forecasting. However, RNNs only have one hidden vector to store the temporal information, which limits their ability to capture long-term dependencies [8]. Moreover, RNNs suffer from the well-known vanishing gradient problem [17]. As a remedy, Hochreiter and Schmidhuber [8] introduced LSTMs, which contain two hidden vectors and an additional memory vector to store information. Thus, LSTMs are more successful in modeling long-term dependences compared to RNNs. Furthermore, ConvLSTM improves the setup of the LSTMs by integrating convolution operations to the input and the hidden state transitions. In addition, the ConvLSTM architecture can take multiple time series as input to produce outputs for each series simultaneously.

We use ConvLSTMs to capture the spatial correlations present in the weather data, through the use of convolution operations. Thus, we incorporate a specific ConvLSTM architecture into our Weather Model to capture the relation between adjacent grids in the spatio-temporal time series. Each ConvLSTM unit in our Weather Model has inputs \( x_t = \{X_1, ..., X_{in}\} \in \mathbb{R}^{M \times N \times L_{in}} \), cell outputs \( S_t = \{S^1_t, ..., S^L_{out}\} \in \mathbb{R}^{M \times N \times L_{out}} \), hidden states \( H_t = \{H^1_t, ..., H^L_{hid}\} \in \mathbb{R}^{M \times N \times L_{hid}} \), and gates \( i_t, f_t, o_t, \) where \( L_{in}, L_{out}, \) and \( L_{hid} \) specify the input, output, and hidden dimensions, respectively. \( x_t \) and \( H_t \) are the results of 2-D convolutions in the input gates shown in Fig. 1(b). The update equations of the gates in ConvLSTM are given as

\[
\begin{align*}
    i_t &= \sigma(W_{xi} \ast x_t + W_{hi} \ast H_{hid,t-1} + b_i) \\
    f_t &= \sigma(W_{xf} \ast x_t + W_{hf} \ast H_{hid,t-1} + b_f) \\
    S_t &= f_t \odot S_{hid,t-1} + i_t \odot \tanh(W_{xs} \ast x_t + W_{hs} \ast H_{hid,t-1} + b_s) \\
    o_t &= \sigma(W_{xo} \ast x_t + W_{ho} \ast H_{hid,t-1} + W_{so} \odot S_t + b_o) \\
    H_t &= o_t \odot \tanh(S_t)
\end{align*}
\]

where \(* \) and \( \odot \) are the convolution and elementwise multiplication operators, respectively. \( W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xs}, W_{hs}, W_{xo}, W_{ho}, W_{so}, b_i, b_f, b_s, b_o \) are the parameters to learn and \( I \) and \( J \) are the height and width of the filters, respectively.

In our Weather Model, we observe the movement patterns in weather features, e.g., temperature, wind, and humidity due to the generic atmospheric circulation, through the use of ConvLSTMs. Here, we regard the different states present in the ConvLSTM units as the hidden representation of the aforementioned movement patterns, where we employ the filters of ConvLSTM to capture these patterns. As illustrated in Fig. 1(a), in order to extend the learning capacity of our Weather Model, we incorporate multiple ConvLSTM units into our framework in a stacked manner.

B. Interpreting the Learned Information From ConvLSTMs

In order to interpret the information learned by the ConvLSTM units, we introduce an encoder–decoder structure and incorporate it into our Weather Model. The encoder part of our model processes the input data through the use of ConvLSTMs and generates a fixed-length context vector that captures all the relevant information contained in the input data. Next, the decoder part interprets the knowledge of the context vector and generates the output data, which is the predicted target sequence. Here, we additionally integrate an attention mechanism into the encoder architecture and a context matcher mechanism into the decoder architecture to further boost their performances, which we support through an extensive set of simulations. Next, we thoroughly explain our encoder and decoder architectures.

1) Encoder and Attention Mechanism: We introduce an encoder structure to model the spatio-temporal sequences...
as shown in Fig. 1(a). For $L$ number of spatio-temporal input sequences $\mathcal{X}_t = \{\mathbf{X}_1^t, \ldots, \mathbf{X}_L^t\}$, our encoder learns the following mapping:

$$\mathcal{H}_t, \mathcal{S}_t = f(\mathcal{X}_t, \mathcal{H}_{t-1}, \mathcal{S}_{t-1})$$

(3)

where $\mathcal{H}_t, \mathcal{S}_t \in \mathbb{R}^{M \times N \times L_{\text{hid}}}$ are the hidden states of the encoder at time $t$ and $\mathcal{X}_t$ is the output of the attention mechanism. Here, $\mathcal{S}_t$ is omitted from Fig. 1(a) and (c) for clarity, in which $\mathcal{H}_t$ and $D_t$ represent the hidden states of our encoder and decoder, respectively. The function $f$ in (3) represents the stacked ConvLSTM units of the encoder. Our encoder recursively performs mapping by using the output states of the previous time steps. We illustrate the unrolled view of the encoder in Fig. 1(c) by showing operations at each time step. In each layer, ConvLSTM units encode the input spatio-temporal time series by referring to the hidden state at the previous time step. As given in Fig. 1(a), after the input layer, each subsequent layer takes the previous layer’s output as input to their ConvLSTM unit. Each layer has unique hidden states $\mathcal{H}_t^l$ and $\mathcal{S}_t^l$, where $k$ represents the layer index. After the input sequence passes through all of the encoder layers, the hidden states $\mathcal{H}_t^l$ and $\mathcal{S}_t^l$ at each encoder layer are passed to the context matcher mechanism, which is the final step of the encoding operation.

To further boost the accuracy of the encoder model, we introduce a specific attention mechanism for this particular spatio-temporal problem. Our attention mechanism helps the encoder extract relevant information from the spatio-temporal time series at each time step by referring to the hidden state of the previous encoder. As shown in Fig. 1(a), for each time step $t$, the attention mechanism takes the input features of the spatio-temporal time series $\mathcal{X}_t = \{\mathbf{X}_1^t, \ldots, \mathbf{X}_j^t, \ldots, \mathbf{X}_L^t\}$ and the encoder’s first layer of the previous hidden state $\mathcal{H}_{t-1}$ to calculate the attention weights for each feature $l$ given as

$$E_l = \mathcal{V} \tanh(\mathcal{W}_E \star \mathcal{H}_{t-1}^l + \mathcal{U} \star \mathbf{X}_l^t + \mathbf{b}_E)$$

(4)

where $\mathcal{W}_E \in \mathbb{R}^{I \times J \times L_{\text{hid}}} \times \mathbb{R}^{L \times L_{\text{hid}}}$, $\mathcal{U} \in \mathbb{R}^{I \times J \times L_{\text{hid}} \times \mathbb{R}^{L_{\text{hid}}}}$, $\mathcal{V} \in \mathbb{R}^{I \times J \times L_{\text{hid}} \times \mathbb{R}^{L_{\text{hid}}}}$, and $\mathbf{b}_E \in \mathbb{R}^{I \times J \times L_{\text{hid}}}$ are the parameters to learn, $L_{\text{att}}$ specifies the attention dimension size, and $I$ and $J$ are the height and width of the filters, respectively. The output dimension of (4) is $M \times N \times 1$ and $E_l^l = \{e_{1,1}^l, \ldots, e_{i,j}^l, \ldots, e_{i,j}^l\}_{i,j,M,N}$ is called the energy matrix for the feature $l$ at time step $t$. We perform softmax operation on the energy matrices of each feature to obtain the attention matrix $A_l^l = \{a_{1,1}^l, \ldots, a_{i,j}^l, \ldots, a_{i,j}^l\}_{i,j,M,N}$, where the softmax operation at each entry of the attention matrix is given as

$$a_{i,j}^l = \frac{\exp(e_{i,j}^l)}{\sum_l^L \exp(e_{i,j}^l)}.$$  

(5)

Next, we obtain the weighted input spatio-temporal time series by taking the Hadamard product of each feature matrix with the corresponding attention matrix

$$\hat{\mathbf{X}}_t^l = \{A_l^l \odot \mathbf{X}_1^l, \ldots, A_l^l \odot \mathbf{X}_L^l\}$$

(6)

where $A_l^l, \hat{\mathbf{X}}_t^l \in \mathbb{R}^{M \times N}$. Each cell in $A_l^l$ represents the attention value assigned to the respective features of that particular cell. Through the integration of the attention mechanism, our encoder is able to selectively focus on different spatio-temporal time series at each time step. Note that the *operation in (4) corresponds to 2-D convolution. We integrate convolutional operations into the attention mechanism as the fully connected methods require 2-D inputs. In essence, without the convolution operations, we would have to flatten the input tensor and, hence, lose the information regarding the spatial covariances that we capture through our ConvLSTM units.

2) Decoder and Context Matcher Mechanism: We present a specific decoder architecture to extract the information encoded within the context vector produced by the encoder. As illustrated in Fig. 1(a), our decoder consists of stacked ConvLSTM units, the same as the encoder. To prevent performance loss given long-length inputs, we additionally incorporate a context matcher mechanism. This mechanism aggregates the hidden states of the encoder layers, given by $\mathcal{H}_t + \cdots + \mathcal{H}_t$. Thus, we can increase the length of the gradient flow up to the first time step. As shown in Fig. 1(a), we use symmetry in our architecture and set the number of layers in the encoder and the decoder to be the same. The encoder’s hidden state dimensions decrease as the number of layers increases and vice versa for the decoder. We describe our context matcher mechanism as

$$D_{t}^{K-k+1} = \frac{1}{t-i-t_{in}} \sum_{i=t_{in}}^{t} \mathcal{H}_{t}^{i}$$

(7)

where $D_{t}^{K-k+1} \in \mathbb{R}^{M \times N \times L_{\text{hid}}}$ is the decoder hidden state, $K$ is the total number of layers, and $k \in \{1, \ldots, K\}$. As shown in Fig. 1(a), the context matcher sums the hidden states and reverses the order of the states. Here, we reverse the matching of the hidden states of the encoder’s layers to increase the performance of the mechanism. Next, the produced state $D_{t}^{1}$ is passed to the decoder’s corresponding layers and the input $Y_t$ passes the first layer of the decoder, as shown in Fig. 1(a). The first layer produces the next hidden state $D_{t+1}^{1}$ and passes it to the next layer. The decoder’s final layer produces the input to the output convolutions, $D_{t+1}^{K}$, as shown in Fig. 1(a). The output convolutions produce the prediction $\hat{Y}_{t+1}$ for the time step $t + 1$. Through this approach, we can utilize higher dimensions in the initial layer of the encoder, thereby achieving predictions with any desired number of dimensions. The recursive prediction of the decoder is given in Fig. 1(c). The decoder uses the predictions produced in the previous time step, allowing us to set the forecast horizon to any value.

C. Evaluation Metrics

During our experiments, we use the benchmark metrics given in [23] for fair comparison. Following their application, we also weigh our evaluation metrics with a predefined latitude weight factor. Since the distance between longitudes decreases as we move toward the poles, the area of each cell in the grid is not equal. The cells that are close to the poles have less area than the cells that are close to the Equator. Thus, we weigh our metrics according to the cell sizes. For the $j$th latitude, the weight factor is given as

$$l(j) = \frac{\cos(\text{lat}(j))}{\sum_{j=1}^{N} \cos(\text{lat}(j))}$$

(8)
where \( N \) represents the number of latitudes and \( \text{lat}(j) \) represents the \( j \)th latitude. Our primary metric is the latitude-weighted root mean square error (RMSE), since it reflects our loss function in (1) and minimizes the effect of the outliers. We define the latitude-weighted RMSE as

\[
\text{RMSE} = \frac{1}{T_{\text{out}}} \sum_{t=T_n+1}^{T_n+T_{\text{out}}} \left[ \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} l(j) \left( \hat{y}_{i,j,t} - y_{i,j,t} \right)^2 \right]^{1/2}
\]

where \( T_{\text{out}} \) represents the output sequence length. As our second evaluation metric, we use latitude-weighted mean absolute error (MAE), given as

\[
\text{MAE} = \frac{1}{T_{\text{out}}} \sum_{t=T_n+1}^{T_n+T_{\text{out}}} \left[ \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} l(j) | \hat{y}_{i,j,t} - y_{i,j,t} | \right].
\]

As our third evaluation metric, we use latitude-weighted mean absolute percentage error (MAPE), given as

\[
\text{MAPE} = \frac{1}{T_{\text{out}}} \sum_{t=T_n+1}^{T_n+T_{\text{out}}} \left[ \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} l(j) \frac{\hat{y}_{i,j,t} - y_{i,j,t}}{y_{i,j,t}} \right].
\]

Finally, we choose the latitude-weighted anomaly correlation coefficient (ACC) as our last metric, defined as

\[
\text{ACC} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} l(j) \hat{y}_{i,j,t} - \hat{y}_{i,j} \hat{y}_{i,j,t} \hat{y}_{i,j,t} \hat{y}_{i,j,t}}{\left( \sum_{i=1}^{M} \sum_{j=1}^{N} l(j) \left( \hat{y}_{i,j,t} \right)^2 \right)^{1/2}}
\]

where \( \hat{y} \) denotes the difference to the climatology, \( \hat{y}_{i,j,t} = y_{i,j,t} - \hat{y}_{i,j} \), where \( \hat{y}_{i,j} \) represents the climatology and is defined as

\[
\hat{y}_{i,j} = \frac{1}{T_{\text{out}}} \sum_{t} y_{i,j,t}.
\]

The ACC metric takes values between +1 and −1, where positive values show that the forecast anomalies have represented the observed anomalies successfully.

D. Prediction Methods

In our simulations, we forecast multiple time steps ahead of the target endogenous series. For example, if the temporal resolution is hourly and we want to forecast for the three-day interval, we predict for the following 72 steps. We obtain these predictions in three methods. In our first method, sequential prediction, we set the output length \( T_{\text{out}} \) to 72 in our model, then the decoder produces the output for each time step \( t > T_n \). In our second method, iterative prediction, we set the output length \( T_{\text{out}} \) to 6, then use the predictions as inputs for the next 6 h and repeat this process for 12 iterations. However, if the model uses four exogenous series as input, we additionally need to predict the four series at each iteration because we assume that we also do not know the future values of the given exogenous series. In our third method, direct prediction, we directly perform forecasts for the \( n \)th time step. For example, if we want to predict only the 24th step of the endogenous series, we set the temporal resolution to 24 h and predict for one step, i.e., \( T_{\text{out}} = 1 \). In our experiments, we use all three prediction methods and compare their performance on two different datasets. The following section focuses on the datasets that we employ and our experimental results.

IV. Experiments

We perform two sets of experiments using two different datasets. In the first setup, we assess our model on a high spatial resolution dataset covering Turkey by comparing it with the state-of-the-art deep learning models. In the second setup, we compare our model performance with the state-of-the-art deep learning and physical models on a benchmark dataset called WeatherBench [23].

A. Data Description

1) Weather Dataset: Our first dataset is the ERA5 hourly data on pressure levels [46], where the temporal resolution of the data is 3 h, the spatial resolution is 30 kilometers, and the data belongs to the 100 hPa atm pressure level. The grid dimensions of the data are \((61 \times 121)\). Our data capture the dates between 2000 and 2001, where the spatial range is \(30^\circ-45^\circ\) in latitude and \(20^\circ-50^\circ\) longitude covering the Mediterranean and Black Sea Coasts of Turkey. In Table I, we illustrate the weather features available for use in the dataset. Among these features, we select the temperature as our endogenous series and the remaining features as our exogenous series. Thus, our task is to find the next temperature values using the previous values of the weather features. We split the dataset into the train, validation, and test sets, with 0.8, 0.1, and 0.1 split ratios, respectively.

2) Benchmark Dataset: Our second dataset is WeatherBench, which is introduced and explained in [23]. The grid size of the data is \((32 \times 64)\), and the temporal resolution is hourly. The 850 hPa temperature feature is selected as our endogenous series, whereas the remaining 18 features are selected as our exogenous features. We obtain direct and iterative forecasts for three to five days of lead time. For the training and validation, we use the data between 2015 and the end of 2016 and separate the data between 2017 and 2018 as the test dataset.

B. Baseline Models

We select five deep learning models to compare their performances with our Weather Model.

1) ConvLSTM: We follow the forecasting network in [7], which involves an encoder–decoder structure. We select this model as our ablation experiment, as we aim to achieve superior performance using our Weather Model with attention, context matcher, and output convolutions. The encoder consists of three layers of ConvLSTM units. 1, 16, 32 and 32, 16, 1 are the number of hidden dimensions in the encoder and the decoder layers, respectively.

2) U-net: We implement the given model architecture due to its success in the recent weather forecasting literature [47]. The model takes a sequence of spatio-temporal temperature time series as the input and predicts their
output sequences. The architecture consists of four downsampling and four upsampling layers.

3) TrajGRU: TrajGRUs have shown slightly better performance than ConvLSTMs in the recent literature [11]. Thus, we added this model to compare its performance with ConvLSTM and our Weather Model. The encoder–decoder structure is the same as the baseline ConvLSTM, and we set the number of links to 1.

4) LSTM: We integrate LSTMs into our experimental setup in order to compare the performances of the ConvLSTMs with vanilla LSTMs. We flatten the grid at each time step to create \((T, MN)\) input matrices to feed the LSTMs. We set the hidden dimension to 256. After the LSTM produces \((T, 256)\) output matrices, we pass them to an FC-NN with \((256, MN)\) matrix size. After reshaping the outputs to the grid size, we produce the outputs for the given spatio-temporal time series.

5) Simple Moving Average: We implement a simple but effective statistical model called the simple moving average (SMA). In this model, we predict the value of the series at time \(t\) by taking the weighted average of the spatio-temporal input series values with window length \(T\) [48]. The averaging operation can be performed with equal weights, or we can give a higher value to the recent values of the series, such as \(t-1, \ldots, t-7\). To further boost the performance of the model, we implement an FC-NN to find the optimal weights for each time series. At each epoch, we initialize the weights with a uniform random variable where the range is \([0, 1]\) and optimize them using the FC-NN.

C. Hyperparameter Selection

To select the best performing parameters for each model to make a fair comparison, we split the datasets into train, validation, and test periods. We train our model on the train set and validate it on the validation set to find the best collection of parameters. We then train the models on the train and validation sets combined using the best collection of parameters. Lastly, we test our model on the test data to determine its performance for comparison.

We use the Adam optimizer with 0 weight decay in each experiment and select the best performing learning rate from \([0.01, 0.001, 0.0005, 0.00001]\). We set the layer size equal for each baseline model and choose the best performing hidden dimension size from \([16, 32, 64]\).

We performed autocorrelation (ACF) and partial autocorrelation analysis (PACF) on temperature values of a randomly selected cell, as shown in Fig. 2. We observed a high correlation until lag 20 with the ACF. In addition, with the PACF, we can find the direct relationship between the observation and its lagged values by removing correlations due to the terms at shorter lags [49]. As illustrated in Fig. 2, we observe high correlation in the first two lags and meaningful correlations until the tenth lag. Therefore, we selected ten time steps of input data with 3 h of frequency and predicted ten time steps ahead.

In addition, input series have different units and scales, hence requiring normalization. However, the dataset is massive, and we need high RAM capacity to normalize a dataset of such size. Thus, we instead apply min–max normalization to the batches in an adaptive manner.

D. Performance Analysis and Results

In the first setup of experiments, we evaluate the performance of our model on the high-resolution dataset by comparing it with the baseline deep learning models [7], [11], [47]. All the models produce predictions using the sequential method introduced in Section III-D, where the output sequence length is set to 5. We illustrate both numerical and visual results to better assess the model performances.

The numerical results of the experiment are illustrated in Table II. We performed a two-tailed t-test with a \(p\)-value of 0.05 comparing ConvLSTM results with our Weather Model, where the Weather Model is shown to perform significantly better in all metrics. Furthermore, the Weather Model outperforms all baseline deep learning models, including TrajGRU and U-Net. When we compare each baseline method, we observe that TrajGRU shows similar performance to ConvLSTM. Furthermore, the models that involve convolution operations, e.g., U-Net, ConvLSTM, and TrajGRU, outperform fully connected models, e.g., LSTM and SMA. Thus, our experiments demonstrate the necessity of using convolution operations to accurately model weather movements, as the

| Long Name         | Short Name | Description                                                                 | Unit            |
|-------------------|------------|------------------------------------------------------------------------------|-----------------|
| geopotential      | z          | The gravitational potential energy of a unit mass                            | \(m^2 s^{-2}\)  |
| potential vorticity| pv         | The measure of the capacity for air to rotate in the atmosphere              | \(K m^2 kg^{-1} s^{-1}\) |
| relative humidity | r          | The water vapor pressure as a percentage of the value at which the air becomes saturated | %               |
| specific humidity | s          | The mass of water vapour per kilogram of moist air                           | \(kg kg^{-1}\)  |
| temperature       | t          | Temperature                                                                  | \(K\)           |
| u-wind            | u          | The eastward component of the wind                                           | \(ms^{-1}\)     |
| v-wind            | v          | The northward component of the wind                                          | \(ms^{-1}\)     |
| vertical velocity | w          | The speed of air motion in the upward or downward direction                 | \(Pa s^{-1}\)   |

| Table I |

| Description and Corresponding Units of the Features Present in the ERA5 Dataset |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
convolution operation crucially impacts the models’ ability to capture spatial covariances in the data.

To visually analyze the spatial and temporal performances, we illustrate the predicted and ground truth sequences in Fig. 3(a). Observing the changes between the prediction steps, we observe that our Weather Model successfully captures temperature fluctuations in up to three steps. As the prediction interval increases, there can be seen some performance decreases, as the model starts to focus on the stationary parts of the sequence. For a more comprehensive study and fair comparison, we provide the predicted sequences of our Weather Model and the benchmark models in Fig. 3(d). Even though all three models, ConvLSTM, U-Net, and our Weather Model, use a type of convolution-based LSTM network as the building block, the state-of-the-art models tend to deteriorate in performance more rapidly compared to our Weather Model. In our Weather Model, the context matcher mechanism helps to maintain long-term dependences by satisfying the gradient flow from the first data sample to the last.

Furthermore, with the help of attention weights shown in Fig. 3(c) and the input series shown in Fig. 3(b), we show that our model attends to the movements in the input features and produces the next hidden state by referring to the temperature at the previous step, demonstrating the success of the attention mechanism. As seen, the attention mechanism boosts our Weather Model’s performance to selectively focus on different parts of the input sequence. At each time step, the model focuses on those cells in each feature it finds significant for advancing to the next hidden state. This determination is guided by the input, which corresponds to the temperature value from the previous step. We observe that the model pays attention to the moving parts in each feature, e.g., potential vorticity and geopotential. Hence, with the help of the attention and context matcher mechanisms, our Weather Model provides relatively better results than its convolution-based alternatives.

In the second setup of experiments, we evaluate the performance of our Weather Model on the WeatherBench dataset by comparing its performance with the physical and baseline models. Additionally, we compare our results with the CNN model given in WeatherBench. We obtain two different results for each baseline model, one for the sequential and one for the iterative method described in Section III-D. In both methods, we aim to forecast up to 72 h. Thus, we set the output sequence length to 72 and 6 in sequential and iterative methods, respectively. We show the results in Table III. Despite the U-net model obtaining the best scores in the sequential method, our Weather Model, trained iteratively, obtains the best scores among all deep learning models. We show the hourly performance of the models in Fig. 4, where the error growth is seen to be logarithmic in RMSE and MAE and linear in ACC metrics. On the other hand, the error growth of the CNN model in [23] is exponential. Hence, our Weather Model solves the exponential error growth problem present in [23].

We also perform direct forecasts with our Weather Model and compare its performance with the direct CNN model. We set the temporal resolution to 24 h and the output sequence length to 3. The CNN model performs better than the Weather Model in the direct method. To further increase the performance of the direct Weather Model, we set the temporal resolution to 72 h and predict a single step at a time. However, this method performs worse than the previous method. After we observed the input sequence in both direct setups, we concluded that the spatial movements are lost within significant temporal gaps. Consequently, we conclude

---

**TABLE II**

| Model Name | RMSE     | MAE     | MAPE      |
|------------|----------|---------|-----------|
| Weather Model | 1.22115  | 0.88872 | 0.00394   |
| U-net      | 1.33032  | 0.97636 | 0.00434   |
| ConvLSTM   | 1.59211  | 1.18715 | 0.00527   |
| TrajGRU    | 1.60205  | 1.21569 | 0.00540   |
| SMA        | 1.90273  | 1.49958 | 0.00666   |
| LSTM       | 1.99607  | 1.62078 | 0.0072    |

We illustrate the performance of the models on the high-resolution dataset. The best results for each metric are given in bold.

---

Fig. 2. Partial autocorrelation and autocorrelation functions of temperature value for a randomly selected cell. The blue region shows the 95% confidence level, and each lag represents a 3-h difference.
We show the output sequence of (a) our model along with (b) input sequences and (c) attention weights at each time step. At each step, the model focuses on those cells in each feature it finds significant for advancing to the next hidden state. This determination is guided by the input, which corresponds to the temperature value from the previous step. We observe that the model pays attention to the moving parts in each feature, e.g., potential vorticity and geopotential. When we compare the predicted and ground truth series in (a), we observe that the model is successful in the first step. After the first step, the model can forecast only the stationary parts, such as the regions that stay cold or hot. To compare our Weather Model with the benchmark models, we also provide the predicted sequences of the baseline models in (d) given arbitrary time.
Fig. 3. (Continued.) We show the output sequence of (a) our model along with (b) input sequences and (c) attention weights at each time step. At each step, the model focuses on those cells in each feature it finds significant for advancing to the next hidden state. This determination is guided by the input, which corresponds to the temperature value from the previous step. We observe that the model pays attention to the moving parts in each feature, e.g., potential vorticity and geopotential. When we compare the predicted and ground truth series in (a), we observe that the model is successful in the first step. After the first step, the model can forecast only the stationary parts, such as the regions that stay cold or hot. To compare our Weather Model with the benchmark models, we also provide the predicted sequences of the baseline models in (d) given arbitrary time.

Fig. 4. Evaluation metrics comparison of baseline models for 72 h of the forecast. We show the best performing method for each model. We also show the results of the direct Weather Model for the given 72 h of forecast.

that the performance of our Weather Model is contingent upon the spatial movements present in the data.

Hence, in this set of experiments, we extensively compare our Weather Model with various baseline learning and physical models. We also satisfy a more fair and exhaustive comparison than other studies in the literature, as we demonstrate the performance of deep learning models under all sequential, iterative, and direct prediction settings. On the other hand, even though our Weather Model performs better than IFS T42 in the RMSE metric, the physical models significantly outperform all given deep learning models. The reason is that the physical models can simulate the atmosphere better for long forecasts. On the contrary, our Weather Model demonstrates high performance in shorter periods.
V. CONCLUSION

We studied the problem of spatio-temporal weather forecasting and introduced a novel deep learning architecture. We introduced a complete model architecture built on stacked ConvLSTM units with attention and context matcher mechanisms to extend the classical encoder–decoder method. We evaluated our model on high-resolution and benchmark datasets and the results show that our model achieves statistically significant performance gains over the state-of-the-art baseline models. We also showed that the error growth of our model on the long forecasts is logarithmic, while it is exponential in the benchmark approach. In addition, we also compared our model with high-resolution physical models, and the physical models show superior performance due to their accurate long forecasts, with a tradeoff of computation time and resources.

ACKNOWLEDGMENT

The authors would like to thank Turkish State Meteorological Services for providing their extensive experience in weather forecasting.

REFERENCES

[1] J. Figa and A. Stoffelen, “On the assimilation of Ku-band scatterometer winds for weather analysis and forecasting,” IEEE Trans. Geosci. Remote Sens., vol. 38, no. 4, pp. 1893–1902, Jul. 2000.

[2] D. Brunet and D. Sills, “A generalized distance transform: Theory and applications to weather analysis and forecasting,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 3, pp. 1752–1764, Mar. 2017.

[3] P. Bauer, A. Thorpe, and G. Brunet, “The quiet revolution of numerical weather prediction,” Nature, vol. 525, no. 7567, pp. 47–55, Sep. 2015.

[4] P. Lynch, “The origins of computer weather prediction and climate modeling,” J. Comput. Phys., vol. 227, no. 7, pp. 3431–3444, Mar. 2008.

[5] J. K. Lazo, R. E. Morss, and J. L. Demuth, “300 billion served: Sources, perceptions, uses, and values of weather forecasts,” Bull. Amer. Meteorol. Soc., vol. 90, no. 6, pp. 785–798, Jun. 2009. [Online]. Available: http://www.jstor.org/stable/26220993.

[6] E. R. Rodrigues, I. Oliveira, R. Cunha, and M. Netto, “DeepDownscale: A deep learning strategy for high-resolution weather forecast,” in Proc. IEEE 14th Int. Conf. e-Soc. (e-Soc’), Oct. 2018, pp. 415–422.

[7] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 802–810.

[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[9] Z. Ma, H. Zhang, and J. Liu, “PrecipLSTM: A meteorological spatiotemporal LSTM for precipitation nowcasting,” IEEE Trans. Geosci. Remote Sens., vol. 60, 2022, Art. no. 4109108, doi: 10.1109/TGRS.2022.319222.

[10] L. Han, H. Liang, H. Chen, W. Zhang, and Y. Ge, “Convective precipitation nowcasting using U-Net model,” IEEE Trans. Geosci. Remote Sens., vol. 60, 2022, Art. no. 4103508, doi: 10.1109/TGRS.2021.3100847.

[11] X. Shi et al., “Deep learning for precipitation nowcasting: A benchmark and a new model,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5617–5627.

[12] L. Han, J. Sun, and W. Zhang, “Convolutional neural network for convective storm nowcasting using 3-D Doppler weather radar data,” IEEE Trans. Geosci. Remote Sens., vol. 58, no. 2, pp. 1487–1495, Feb. 2020, doi: 10.1109/TGRS.2019.2948070.

[13] Y. V. Wang et al., “Relative importance of radar variables for nowcasting heavy rainfall: A machine learning approach,” IEEE Trans. Geosci. Remote Sens., vol. 61, 2023, Art. no. 4100314, doi: 10.1109/TGRS.2022.3231125.

[14] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. Courville, “Modulating early visual processing by language,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 6594–6604.

[15] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual reasoning with a general conditioning layer,” in Proc. AAAI Conf. Artif. Intell., vol. 32, 2018, pp. 3942–3951.

[16] X. Wang, K. Yu, C. Dong, and C. Change Loy, “Recovering realistic texture in image super-resolution by deep spatial feature transform,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 606–615.

[17] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, “A dual-stage attention-based recurrent neural network for time series prediction,” 2017, arXiv:1704.02971.

[18] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2014, arXiv:1409.0473.

[19] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2017, pp. 5998–6008.

[20] L. Zhang, G. Zhu, L. Mei, P. Shen, S. A. A. Shah, and M. Bennamoun, “Attention in convolutional LSTM for gesture recognition,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, 2018, pp. 1953–1962.

[21] S. Agrawal, L. Barrington, C. Bromberg, J. Burge, C. Gazer, and J. Hickey, “Machine learning for precipitation nowcasting from radar images,” 2019, arXiv:1912.12132.

[22] Q.-K. Tran and S.-K. Song, “Computer vision in precipitation nowcasting: Applying image quality assessment metrics for training deep neural networks,” Atmosphere, vol. 10, no. 5, p. 244, May 2019.

[23] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerer, “WeatherBench: A benchmark data set for data-driven weather forecasting,” J. Adv. Model. Earth Syst., vol. 12, no. 11, Nov. 2020, Art. no. e2020MS002203.

[24] J. N. K. Liu, Y. Hu, Y. He, P. W. Chan, and L. Lai, “Deep neural network modeling for big data weather forecasting,” in Information Granularity, Big Data, and Computational Intelligence. Cham, Switzerland: Springer, 2015, pp. 389–408.

[25] M. Hossain, B. Rekabdar, S. J. Louis, and S. Dascu, “Forecasting the weather of Nevada: A deep learning approach,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2015, pp. 1–6.

[26] J. Lee, C.-G. Kim, J. Lee, N. Kim, and H. Kim, “Application of artificial neural networks to rainfall forecasting in the Geum river basin, Korea,” Water, vol. 10, no. 10, p. 1448, Oct. 2018.

[27] B. Kanigor and A. G. Salman, “Recurrent gradient descent adaptive learning rate and momentum neural network for rainfall forecasting,” in Proc. Int. Seminar Appl. Technol. Inf. Commun. (ISemantic), Aug. 2016, pp. 802–810.

[28] M. Akram and C. El, “Sequence to sequence weather forecasting with long short-term memory recurrent neural networks,” Int. J. Comput. Appl., vol. 143, no. 11, pp. 7–11, Jun. 2016.

[29] S. Poornima and M. Pushpalatha, “Prediction of rainfall using intensified LSTM based recurrent neural network with weighted linear units,” Atmosphere, vol. 10, no. 11, p. 668, Oct. 2019.
[30] B. Wang et al., “Deep uncertainty quantification: A machine learning approach for weather forecasting,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2019, pp. 2087–2095.

[31] P. Hewage, M. Trovati, E. Pereira, and A. Behera, “Deep learning-based effective fine-grained weather forecasting model,” Pattern Anal. Appl., vol. 24, no. 1, pp. 343–366, Feb. 2021.

[32] J. G. Powers et al., “The weather research and forecasting model: Overview, system efforts, and future directions,” Bull. Amer. Meteorol. Soc., vol. 98, no. 8, pp. 1717–1737, Aug. 2017.

[33] W.-C. Woo and W.-K. Wong, “Operational application of optical flow techniques to radar-based rainfall nowcasting,” Atmosphere, vol. 8, no. 3, p. 48, Feb. 2017.

[34] A. Heye, K. Venkatesan, and J. Cain, “Precipitation nowcasting: Leveraging deep recurrent convolutional neural networks,” in Proc. Cray User Group (CUG), 2017.

[35] J. R. Jing, Q. Li, X. Y. Ding, N. L. Sun, R. Tang, and Y. L. Cai, “AENN: A generative adversarial neural network for weather radar echo extrapolation,” Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci., pp. 89–94, Oct. 2019.

[36] K.-S. Kim, J.-B. Lee, M.-I. Roh, K.-M. Han, and G.-H. Lee, “Prediction of ocean weather based on denoising AutoEncoder and convolutional LSTM,” J. Mar. Sci. Eng., vol. 8, no. 10, p. 805, Oct. 2020.

[37] 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. Appl., vol. 2, Dec. 2020, Art. no. 100007.

[38] C.-A. Diaconu, S. Saha, S. Günnemann, and X. Xiang Zhu, “Understanding the role of weather data for Earth surface forecasting using a ConvLSTM-based model,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2022, pp. 1361–1370.

[39] P. Hewage et al., “Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station,” Soft Comput., vol. 24, no. 21, pp. 16453–16482, Nov. 2020.

[40] R. Castro, Y. M. Souto, E. Ogasawara, F. Porto, and E. Bezerra, “STConv2S: Spatiotemporal convolutional sequence to sequence network for weather forecasting,” Neurocomputing, vol. 426, pp. 285–298, Feb. 2021.

[41] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, “An end-to-end spatio-temporal attention model for human action recognition from skeleton data,” in Proc. AAAI Conf. Artif. Intell., vol. 31, 2017, pp. 4263–4270.

[42] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, “GeoMAN: Multi-level attention networks for geo-sensory time series prediction,” in Proc. 27th Int. Joint Conf. Artif. Intell., Jul. 2018, pp. 3428–3434.

[43] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, “Revisiting spatial–temporal similarity: A deep learning framework for traffic prediction,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 5668–5675.

[44] I. A. Abdellouaoui and S. Mehrkanoon, “Deep multi-stations weather forecasting: Explainable recurrent convolutional neural networks,” 2020, arXiv:2009.11239.

[45] J. Elman, “Finding structure in time,” Cogn. Sci., vol. 14, no. 2, pp. 179–211, Jun. 1990.

[46] H. Hersbach et al., “The ERA5 global reanalysis,” Quart. J. Royal Meteorol. Soc., vol. 146, no. 730, pp. 1999–2049, 2020.

[47] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net: Learning dense volumetric segmentation from sparse annotation,” in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv., Athens, Greece. Cham, Switzerland: Springer, Oct. 2016, pp. 424–432.

[48] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice. Melbourne, VIC, Australia: OTexts, 2023.

[49] A. V. Metcalfe and P. S. Cowpertwait, Introductory Time Series With R. New York, NY, USA: Springer, 2009.

Selim Furkan Tekin received the B.S. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2021. He is currently pursuing the M.S. degree with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara.

His research interests include few-shot learning, ensemble learning in diverse contexts, including federated learning, natural language processing, and computer vision.

Arda Fazla received the B.S. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2021. He is currently pursuing the M.S. degree with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara.

His research interests include machine learning, optimization, time series prediction, signal processing, and online learning.

Suleyman Serdar Kozat (Senior Member, IEEE) received the B.S. degree (Hons.) from Bilkent University, Ankara, Turkey, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2001 and 2004, respectively.

He is currently a Professor with the Electrical and Electronics Engineering Department, Bilkent University. He has coauthored more than 120 articles in refereed high-impact journals and conference proceedings and holds several patent inventions. His research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering, and machine learning algorithms for signal processing.