CS559 Project Report
Forecasting of Tropical Cyclone Trajectories with Deep Learning

Selim Furkan Tekin
21501391

Fatih İlhan
21401801

***Abstract*—We study forecasting hurricane (tropical cyclone) trajectories using deep learning techniques. Hurricane trajectories exhibit highly complex and nonlinear behavior. Numerous factors such as landscape, atmoshperic effects cause tropical cyclones to follow devious or unwavering routes. We employ a recurrent neural network (RNN)-based deep learning architecture called TrajGRU [1] to capture these complex temporal patterns and perform forecasts. We analyze the performance in terms of kilometers at various hourly resolutions and compare the results with two baseline methods including naive linear predictor and long short-term memory (LSTM) networks [2]. We demonstrate the performance gains and analyze the predictions.**

I. Introduction

A. Preliminaries

A cyclone, hurricane or typhoon is the system of storms which rotate around a low-pressure center with high speeds. Forecasting the trajectory and intensity of hurricanes is crucial to take precautions for the protection of people and property. Since the dynamics of the system is highly complex, nonlinear, and dependent on numerous external factors, conventional methods which mostly rely on statistical forecasting tools perform poorly. Moreover, they can't properly utilize the significant amount of past trajectory data which grows continuously. In this study, we solve this tracking problem using deep learning techniques considering its advantages in complex system modeling and making use of big data.

Neural network-based methods are becoming widely preferred for the time series prediction task thanks to their ability to approximate highly nonlinear and complex functions [3]. To capture the temporal relations in time-series data properly, recurrent neural networks (RNNs) are used in sequential tasks thanks to their ability to exploit temporal behavior. RNNs contain a temporal memory called hidden state to store the past information, which helps them to model time-series more successfully in several different sequential learning tasks [4], [5], [6]. Hence, we consider RNN-based networks to perform time-series prediction.

There are several forms of recurrent neural networks such as gated recurrent units (GRU) and long short-term memory (LSTM) networks in the literature. In addition their spatial extensions such as ConvLSTM [7] are also widely preferred in spatiotemporal prediction tasks. In this study, we use TrajGRU [1] model with several modifications to adjust to this problem. This model was initially proposed for the precipitation prediction tasks, and has shown success thanks to its warping operation, which captures weather events naturally. In this project, we employ TrajGRU for hurricane trajectory prediction task.

B. Related Work

Recently, several deep learning based approaches has been proposed to track cyclone trajectories. [8] clusters the hurricanes using DTW distance, and employ sparse Recurrent Neural Network (sparse-RNN) architecture, in which the weight connections are optimized using a Genetic Algorithm (GA). In [9], a Convolutional Neural Network (CNN) combined with Long-Short Term Memory (LSTM) units is proposed to predict the coordinates of the hurricane center using weather satellite images. Another simple approach proposes to divide the spatial region into cells, and feeding corresponding spatial and temporal features into an LSTM to predict the next cell position of the hurricane center [2]. In [10], fusing past trajectory data and atmosphere images centered at hurricane centers to predict the displacement vector is proposed. A recent study proposed a methodology to predict map of probabilities of being a center of hurricane or not, using two Convolutional LSTM (ConvLSTM) networks one for tracking and other for predicting [11]. All studies has shown applicable results, however their results are presented in different datasets with different measures at different temporal resolutions.

Some studies [11] approaches this problem as a pixel classification task, and tries to predict whether a hurricane center will exist at a pixel or not. We approach this task as a linear regression problem, and penalize bigger prediction errors heavily with mean squared error loss in terms of degrees. Although the previous studies also employ deep learning models including recurrent neural network and CNN-based models, our model enables to capture the warping structure of hurricanes. GRU is also an efficient model, which prevents the gradient problems related to simple RNN and has the gating feature of LSTMs, but with less complexity. We also utilize a early fusion module (a multi-layer perceptron that embeds external features into input images) that incorporates side information into our predictions.

Although the previous studies has worked on different spatial or temporal subsections of the hurricane dataset, we

particularly focus on the years between 1994 and 2020. We consider the North Atlantic basin since the hurricane density is comparably high and economic effects are significantly greater compared to other basins. During evaluation, we do not present the results in terms of degrees due to the fact that each latitutdal/longitudal degree represents a different spatial distance in terms of kilometers. We instead consider the errors in terms of kilometers while comparing different models. This provides a more practical and interpretable approach while eavluating different methods.

C. Contributions

Our contributions are as follows:

- As the first time in the literature, we employ TrajGRU [1] model for hurricane trajectory forecasting. This model uses atmospheric feature images at different pressure levels and handles both spatial and temporal complexity of the hurricane phenomena.
- We also utilize a early fusion module to incorporate external features including distane to land, storm speed, storm direction, sustained wind speed and center pressure into our architecture.
- We compare the performance of our model with two other methods, naive linear predictor and LSTM [2], and demonstrate the gains in terms of prediction error in kilometers.

D. Organization

The organization of this paper is as follows. In Section II, we define the problem of forecasting hurricane trajectories and provide the mathematical notation. Then, we describe and give the details of the implemented model in Section III. We provide the dataset and baseline method details in Section IV-A and IV-B, respectively. We present the experiment results and compare our architecture with other methods in Section IV-D. Finally, we conclude the paper with several remarks in V.

II. Problem Definition

We study the problem of hurricane trajectory forecasting. Our dataset consists of a set of weather images, one-dimensional external features and hurricane trajectories. Hurricane trajectories are represented as $\{\mathbf{Y}_n\}_{n=1}^{N}$, where N is the number of hurricanes, and $\mathbf{Y}_n \in \mathbb{R}^{T\times 2}$ is the matrix of hurricane center locations (latitude and longitudes) for the n th hurricane with T timesteps. Here, each row captures a 3-hour-long timeframe. We denote the weather images corresponding to these hurricanes with $\{\mathbf{X}_n\}_{n=1}^{N}$, where $\mathbf{X}_n \in \mathbb{R}^{T\times 3\times 25\times 25\times 5}$ is the matrix of weather images that consists of athmospheric features measured at three different pressure levels. External features, which we call side information, is denoted as $\{\mathbf{S}_n\}_{n=1}^{N}$, where $\mathbf{S}_n \in \mathbb{R}^{T\times 5}$. Here, each row representes five features of that timeframe (distance to land, storm speed, storm direction, sustained wind speed and center pressure).

Our goal is to design and train a deep learning model, which can be denoted as a complex nonlinear function (f), which takes the past information for a hurricane and forecasts hurricane center locations along the future. We can express the system as follows:

$$\hat{\boldsymbol{y}}_{n,t+1} = f(\mathbf{X}_{n,:t}, \boldsymbol{s}_{n,:t}; \boldsymbol{\theta}), \tag{1}$$

where $\hat{\boldsymbol{y}}_{n,t+1}$ is the hurricane center location prediction for the $t+1$ th timestep of the nth hurricane, and $\mathbf{X}_{n,:t}$ and $\boldsymbol{s}_{n,:t}$ are the weather image tensor and side information vectors until the t th timestep (including). Here, $\boldsymbol{\theta}$ the model weights. We achieve our goal through minimizing the mean squared error loss between $\hat{\boldsymbol{y}}_{n,t+1}$ and $\boldsymbol{y}_{n,t+1}$ over the training set.

III. Model Description

Our model has an encoder-decoder structure with an input and output block. The architecture of the model is shown in Figure 1. In the following sections, we describe the operations performed in each block.

A. Input Block

There are two sources of information that we use in this project. The first source is coming from weather images cropped by centering the hurricanes: $\mathbf{X}_t \in \mathbb{R}^{T\times L\times M\times N\times D}$, where t is the timestep that is $t - W_{in} \leq t < t+1$ and W_{in} is the input window length. We collected D different atmospheric features from L different atmospheric pressure levels for T time steps with spatial resolution of $M \times N$. Since we use 2-D convolutional operations in our model structure, input data is flatten at level dimension to create $\mathbf{X}_t \in \mathbb{R}^{T\times M\times N\times D*L}$. Second source is the vectoral source that belongs to input hurricane $\mathbf{s}_t \in \mathbb{R}^{T\times D^{'}}$ where $D^{'}$ shows the feature dimension. We use this information by passing through a multi-layer-perceptron and broadcasting to form $\mathbf{S}_t \in \mathbb{R}^{T\times M\times N\times D^{'}}$. Finally these two source of information are concateneted to create input $\mathbf{X}_t^{'} \in \mathbb{R}^{T\times M\times N\times D*L+D^{'}}$.

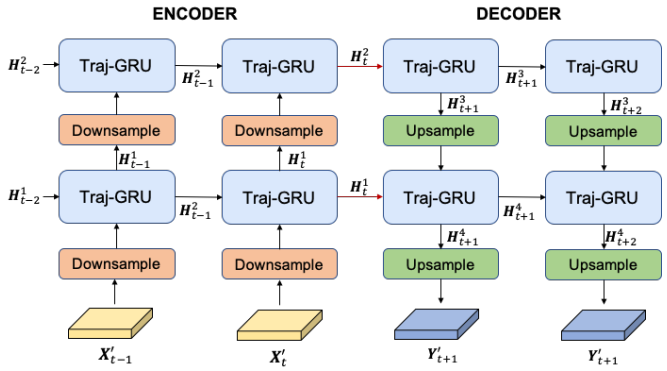
B. Encoder Decoder Block

The encoder-decoder structure allows us to take a sequence of inputs and produce a sequence of outputs. Recurrent units are used as building blocks in this structure which transfers information between timesteps. The main goal of the encoder is to code input sequence to a contex vector and decode the representation into target sequence. For example, $\mathbf{x} = (x_1, x_2, \cdots, x_T)$ is a input sequence where $\mathbf{x} \in \mathbb{R}^T$, the encoder block learns mapping of

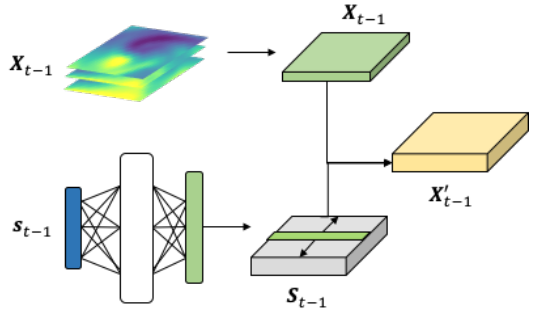
$$\mathbf{h}_t = f(\mathbf{x}, \mathbf{h}_{t-1})$$

where $\mathbf{h}_t \in \mathbb{R}^m$ is the hidden state of the encoder at time t, and m is the size of hidden state. f represents a non-linear function such as RNN[12], LSTM [13], GRU [14], ConvLSTM[11] or TrajGRU [1] as we use in this paper.

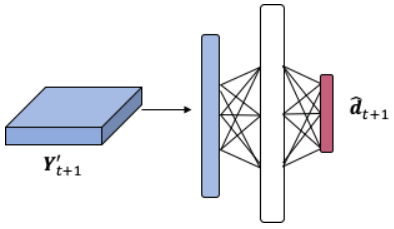
In our problem input sequence is an spatio-temporal sequence. Thus we implemented a model which learns both spatial and temporal correlations. As it is shown in Figure 1a, the



(a) Graph of the encoder-decoder block. The input data comes as a sequence and output is predicted as a sequence. Operations for the two timesteps are shown. The states of the recurrent units are carried between the encoder and the decoder blocks. We use convolution and maxpooling operations for the downsampling operation, and transposed convolution is used for the upsampling operation.



(b) Graph of the input block showing the side information entegration to the input module. The side information $\mathbf{s}_{t-1}$ is fed into an multi layer perceptron. Then, the output vector is broadcasted by repeating along the selected axis. Finally, the broadcasted embedded side information is concataneted with the weather data



(c) Graph of the final block showing the distance vector prediction. The decoder outputs a 2-D tensor. Then tensor is flattened and passed through a multi layer perceptron to predict a two dimensional vector, $\hat{d}_{t+1}$ which is the prediction for the next location of hurricane center.

Fig. 1: Visualisation of the model architecture and operation steps. Data first passes to 1b, then 1a and finally 1c

.

encoder takes an input sequence, $\mathbf{X} = (\mathbf{X'}_1, \mathbf{X'}_2, \cdots, \mathbf{X'}_T)$ and learns mapping of

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

where $\mathbf{H}_t$ is the coded input representation whe f represents the TrajGRU and downsample operations. Then representation passes to the decoder block. In this block, representation is decoded to output tensor Y'_{t+1} with TrajGRU units and upsampling operations.

TrajGRU learns *location-variant* filters between input frames. With this implementation, TrajGRU can learn the motion patterns between time steps. At each time step, TrajGRU takes current input and previous state to generate local neighborhood set for each location at each timestep. The main formulas of TrajGRU are given as follows [1]:

$$\begin{aligned}
&\mathcal{U}_t, \mathcal{V}_t = \gamma(\mathcal{X}_t, \mathcal{H}_{t-1}),\\
&\mathcal{Z}_t = \sigma(\mathcal{W}_{xz} * \mathcal{X}_t + \sum_{l=1}^{L} \mathcal{W}^l_{hz} * \text{warp}(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l})),\\
&\mathcal{R}_t = \sigma(\mathcal{W}_{xr} * \mathcal{X}_t + \sum_{l=1}^{L} \mathcal{W}^l_{hr} * \text{warp}(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l})),\\
&\mathcal{H}'_t = f(\mathcal{W}_{xh} * \mathcal{X}_t + \mathcal{R}_t \circ (\sum_{l=1}^{L} \mathcal{W}^l_{hh} * \text{warp}(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l}))),\\
&\mathcal{H}_t = (1 - \mathcal{Z}_t) \circ \mathcal{H}'_t + \mathcal{Z}_t \circ \mathcal{H}_{t-1}.
\end{aligned} \tag{2}$$

Here, L is the total number of allowed links. $\mathcal{U}_t, \mathcal{V}_t \in \mathbb{R}^{L \times H \times W}$ are the flow fields that store the local connection structure generated by the structure generating network γ. The $\mathcal{W}^l_{hz}, \mathcal{W}^l_{hr}, \mathcal{W}^l_{hh}$ are the weights for projecting the channels, which are implemented by 1×1 convolutions. The $\text{warp}(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l})$ function selects the positions pointed out by $\mathcal{U}_{t,l}, \mathcal{V}_{t,l}$ from $\mathcal{H}_{t-1}$ via the bilinear sampling kernel.

C. Output Block

In this block we predict next location of the hurricane center for each frame. The decoder outputs $\mathbf{Y'_{t+1}}$. Then this tensor is flatten and passed through multi-layer-perceptron for each time step. Finally, the distance vector $\hat{\mathbf{d}}_{t+1}$ is predicted where $\mathbf{d} = (\delta_{latitude}, \delta_{longitude})$ for each frame.

IV. Experiments

A. Datasets

We use IBTrACS Dataset for hurricane trajectories [15]. This dataset contains the hurricane trajectory data since 1842 from multiple meteorology agencies. Every hurricane trajectory has 3-hourly center coordinates, with additional features such as land distance, hurricane type and wind speed. The dataset contains more than 13000 hurricane tracks. The dataset includes samples from different basins, e.g North Atlantic (NA), Eastern North Pacific (EP). Since hurricanes show different characteristics based on different basins, only hurricanes occurred in North Atlantic basin are selected. In addition, NA contains less interpolated features, more samples and is known as the benchmark region. Data includes latitude, longitude, distance to land, maximum sustained wind speed, minimum sea level pressure, storm speed and storm direction features for every 3 hours of hurricane. Data is truncated to years between 1994 and 2020 because of the gaps in selected features. Between selected years there are 426 hurricanes recorded. This is a reasonable number of hurricanes considering other works

[8] [9] [10]. This dataset was directly used by the baseline method.

For weather images, we have collected reanalysis weather data for different pressure levels from ERA Interim Dataset [16]. This dataset provides atmospheric feature images between 1979 and 2020. We cropped 25x25 km images for each timestep such that the hurricanes are centered. We have used 5 different features (divergence, geopotential, u-component of wind, v-component of wind, fraction of cloud cover) from 3 different pressure levels (300, 500 and 1000 hPa), in total 15 features.

In Figure 2, we illustrate some samples of hurricane trajectories. We also visualize some unique hurricane trajectories with their measured storm directions in Figure 3. As shown, they can exhibit quite nonlinear and complex patterns.

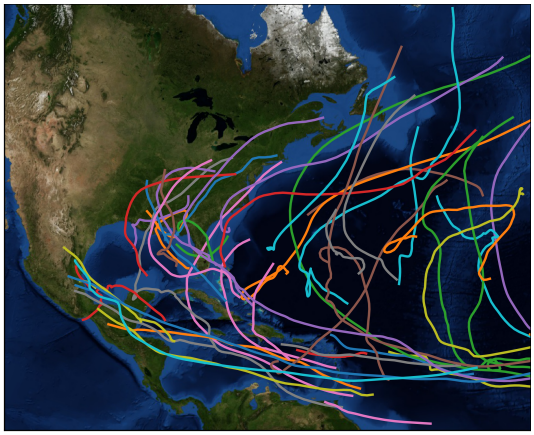


Fig. 2: Samples of hurricane trajectories. Some samples have very nonlinear and complex trajectories.

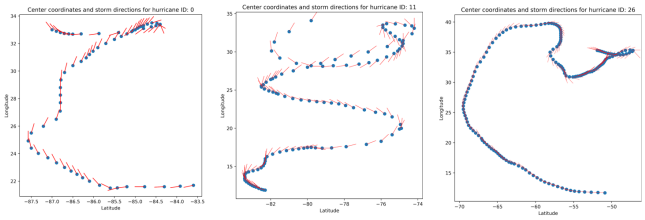


Fig. 3: Samples of hurricane trajectories and directions.

We have also performed exploratory data analysis on hurricane data. In Figure 4 and 5, we provide the histogram of hurricane with respect to various features.

B. Baseline Methods

In addition to our model, we also evaluate two other methods to compare with the performance of our architecture. To this end, we have implement the following baseline methods. The first method is a simple baseline method which adds up the last displacement vector into current location, and the second method is a deep learning-based approach proposed in a recent work [2].

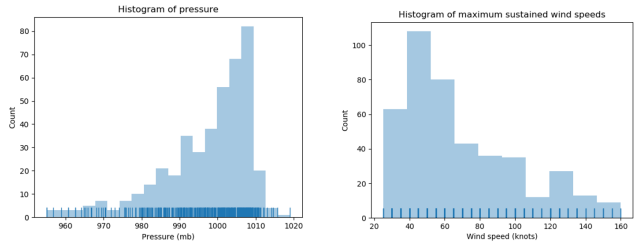


Fig. 4: Histogram of hurricanes with respect to pressure and wind speed

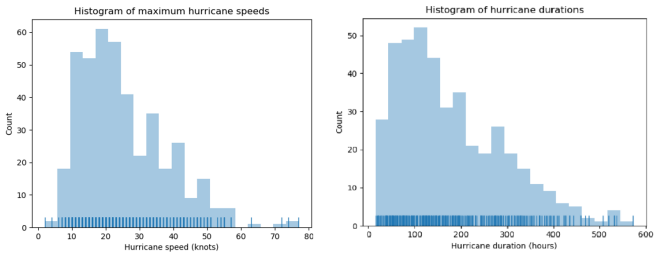


Fig. 5: Histogram of hurricanes with respect to hurricane speed and duration

1) Naive Linear Predictor: The equation that represents the naive linear predictor is below:

$$\hat{x}_t = x_{t-1} + \Delta x_{t-1} \tag{3}$$

$$\Delta x_{t-1} = x_{t-1} - x_{t-2} \tag{4}$$

This predictor adds up the latest displacement vector into current location to predict the next lcoation of the hurricane center. We can also make multi-step predictions by using predictions recurrently as if they are real location values.

2) Long Short-Term Memory (LSTM) Networks: This methodology divides the spatial map of North Atlantic region into a 1x1 degree resolution grid. Then, a multi-layer LSTM network is used to predict the next location of hurricane center on this grid. Model takes a sequence of inputs and predicts the sequence of hurricane location in grid index [2]. This model employs a multi-layer LSTM-based architecture. In addition, next latitude, longitude values of center is predicted by suffering the Mean Square Error (MSE) loss. Figure 6 shows the architecture of this model. In our implementation for the baseline, we were able to reproduce the results claimed in the paper with almost same hyperparameters as described in the paper [2].

C. Implementation Details

Since input features and target values have variety of units we apply normalization. Input and outputs are min-max scaled. We train our model as stateful model, which means states are carried between batches. This way we preserve the information

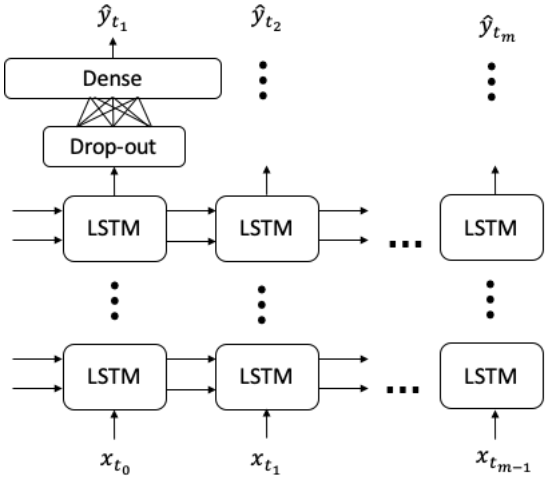


Fig. 6: LSTM network for hurricane trajectory forecasting, proposed in [2]

pass between each time step. However to prevent vanishing gradient problem caused by back propagation through time, we detach states at the begining of each loop. We tuned the hyperparameters with grid search. We took 10 time step length of input and tried to predict next center location of these frames. We split the hurricane data into three sets; test, train and validation and during training we monitor our training and validation error. After five consecutive increase in validation error, we stop the training and took model belonging to five step before as our the best model. Final model is evaluated at the test set and results are recorded.

D. Results

We obtain prediction errors in terms of kilometers for four different forecasting durations, 3, 6, 12 and 24 hours. The results are summarized in the following table:

TABLE I: Forecasting errors for 3, 6, 12 and 24 hours for all methods in terms of kilometers. Experiments are repeated 5 times for LSTM and TrajGRU. We report the average errors.

Method	Forecasting Duration			
	3h	6h	12h	24h
Naive Linear Predictor	**21.3**	67.8	201.3	472.1
LSTM	27.8	83.5	123.3	208.6
TrajGRU	29.4	**66.6**	**112.9**	**189.0**

Except the 3-hour case, TrajGRU performs better that other models. When the forecasting duration increases, the performance improvement becomes more significant. The encoder-decoder structure provides higher accuracy for multi-step long duration predictions. Considering the fact that forecasting hurricanes more earlier is more crucial and useful, our model is more practical even though 3-hour error is higher. Naive linear predictor performs best in 3-hour predictions but it is not really useful since it is not usually possible to take precautions in three hours.

V. Conclusion

In this project, we apply deep learning techniques for hurricane trajectory forecasting. We have modified the TrajGRU model proposed in [1] for precipitation forecasting to this problem. Our architecture takes atmospheric at images at 3-hours resolution as inputs and predicts the location of the hurricane center for the next timesteps. We also use an early side information fusion module to utilize external features including storm speed, direction, distance to land etc. We demonstrate the gains in prediction performance in terms of kilometers and compare our results with naive linear predictor and LSTM networks.

References

[1] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," in *Advances in neural information processing systems*, pp. 5617–5627, 2017.

[2] S. Alemany, J. Beltran, A. Perez, and S. Ganzfried, "Predicting hurricane trajectories using a recurrent neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 468–475, 2019.

[3] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning - a new frontier in artificial intelligence research [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, pp. 13–18, Nov 2010.

[4] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016.

[5] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, no. 34, pp. 1–5, 2017.

[6] W. D. Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Computer Speech & Language*, vol. 30, no. 1, pp. 61 – 98, 2015.

[7] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015.

[8] M. Moradi Kordmahalleh, M. Gorji Sefidmazgi, and A. Homaifar, "A sparse recurrent neural network for trajectory prediction of atlantic hurricanes," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 957–964, 2016.

[9] M. Mudigonda, S. Kim, A. Mahesh, S. Kahou, K. Kashinath, D. Williams, V. Michalski, T. O'Brien, and M. Prabhat, "Segmenting and tracking extreme climate events using neural networks," in *Deep Learning for Physical Sciences (DLPS) Workshop, held with NIPS Conference*, 2017.

[10] S. Giffard-Roisin, M. Yang, G. Charpiat, B. Kégl, and C. Monteleoni, "Deep learning for hurricane track forecasting from aligned spatio-temporal climate datasets," 2018.

[11] S. Kim, H. Kim, J. Lee, S. Yoon, S. E. Kahou, K. Kashinath, and M. Prabhat, "Deep-hurricane-tracker: Tracking and forecasting extreme climate events," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1761–1769, IEEE, 2019.

[12] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[15] K. R. Knapp, M. C. Kruk, D. H. Levinson, H. J. Diamond, and C. J. Neumann, "The international best track archive for climate stewardship (ibtracs) unifying tropical cyclone data," *Bulletin of the American Meteorological Society*, vol. 91, no. 3, pp. 363–376, 2010.

[16] D. P. Dee, S. M. Uppala, A. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. Balmaseda, G. Balsamo, d. P. Bauer, *et al.*, "The era-interim reanalysis: Configuration and performance of the data assimilation system," *Quarterly Journal of the royal meteorological society*, vol. 137, no. 656, pp. 553–597, 2011.