Wind Forecasting Using Hybrid Convolutional Neural Network – Long Short-Term Memory Framework

Andrew Sheerin
Dept. of Civil and Environmental Engineering
University of Rhode Island
Kingston, Rhode Island, United States
andrew_sheerin@uri.edu

I. Project Objective

The objective of this project is to develop a predictive model for short-term wind forecasting in coastal regions using a hybrid deep learning approach. This approach combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to analyze spatial and temporal wind patterns, enabling predictions of wind speed and direction with enhanced accuracy and robustness.

The CNN component extracts critical spatial features from wind measurement data, identifying localized patterns and environmental influences that impact wind dynamics in coastal areas. The LSTM component captures temporal dependencies, such as historical trends, seasonal variations, and short-term fluctuations, ensuring the model reflects both immediate and long-term influences on wind behavior.

The hybrid model is trained using historical coastal wind datasets THAT encompass a range of meteorological conditions and geographic scenarios. Through this training process, the model achieves improved precision and reliability, outperforming conventional forecasting methods. Its applications include optimizing renewable energy operations, improving marine navigation safety, and enhancing preparedness for coastal weather-related challenges.

II. Background

Accurate wind forecast models are of significant value to many industries, including renewable energy, athletics, agriculture, and transportation. These forecasts provide critical information to aid decision-making that could have important economic, environmental, or social impacts. Personal motivation stems from my passion for sailing, a sport that relies solely on the wind, making reliable wind forecasts an asset. There are multiple wind forecast models that are used globally; however, the forecasted wind speed and direction are sometimes inaccurate when compared to observed data. Moreover, each wind model may present discrepancies between each other, making it difficult to determine reliable forecasts. Widely considered as the most accurate model in North America, the High-Resolution Rapid Refresh (HRRR) model considers a 3 km spatial resolution, refreshes every 1-hour, and forecasts up to 36 hours [1].

Weather forecasting methods can be divided into three categories: (1) physical methods, (2) traditional statistical methods, and (3) machine learning-based methods [2]. Physical methods rely on local environmental information, such as temperature or humidity, and typically perform poorly for short-term forecasting. Traditional statistical methods mostly use regression or moving average models but struggle with capturing nonlinear relationships. Machine learning models, particularly neural networks, have been used by researchers because of their powerful ability to solve nonlinear problems [2]. Popular machine learning algorithms for weather forecasting include support vector machines, random forests, convolutional neural networks, and long short-term memory networks [3-7]. Each of these models has demonstrated significant improvements in forecast accuracy in their respective applications. However, hybrid models often outperform single models due to the complexity of weather systems and the limitations inherent to individual techniques [8].

The hybrid CNN-LSTM model provides significant advantages by leveraging the strengths of both algorithms. CNNs excel at extracting spatial features from input data, employing multi-layer convolution and pooling operations to identify patterns in wind behavior that depend on geography, topography, and other spatial factors. These operations allow the model to focus on relevant localized features while reducing computational complexity. In contrast, LSTMs are specialized for capturing temporal dependencies, making them ideal for learning sequential patterns in wind data, such as diurnal cycles, seasonal shifts, and short-term fluctuations. By combining CNN's spatial learning capabilities with LSTM's temporal proficiency, the hybrid model captures both static and dynamic features of wind systems, enabling robust and accurate forecasting [8][9].

Training the CNN-LSTM model involves the use of historical wind measurement datasets, which encompass a variety of meteorological conditions and geographic settings. This ensures that the model learns to generalize effectively across different coastal regions. The hybrid approach addresses the shortcomings of conventional forecasting methods and single-model machine learning approaches, offering improved precision and reliability. The results have wide-ranging applications, including optimizing renewable energy production, enhancing marine navigation safety, and supporting coastal disaster management efforts. This project contributes to advancing environmental modeling by utilizing

the deep learning techniques. By combining the spatial feature extraction capabilities of CNNs and the sequential learning strengths of LSTMs, it offers a scalable and effective solution to the inherent complexities of wind forecasting in dynamic coastal environments.

III. Methodology

The methodology for this project will include data gathering and processing, the development of the CNN-LSTM model, and model evaluation. The following sections will depict the methods required to complete this project.

A. Data Requirements

Historical forecast and observed data from 2021 to 2023 will be collected, focusing on this period due to the significant time and storage requirements for downloading the data. Forecast data will be obtained using the Herbie python package, developed as part of NOAA's Open Data Dissemination Program [10]. Data from the HRRR forecast model will include wind in the U (east-west) and V (north-south) components, temperature, pressure, humidity, precipitation, and cloud cover at a 10 m elevation. For each day, 24 hourly GRIB files will be downloaded, resulting in a total of 26,280 files. Each GRIB file is 10MB, meaning the computational demand is extensive.

Given the intensive data collection, we Unity, a high-performance computing cluster, will be utilized to expedite this the downloading process [11]. To reduce storage needs, the NOAA-developed tool wgrib2 will be used to clip files to the Narragansett Bay extent, reducing each file's size to approximately 12 kB [12]. The Narragansett Bay coastline will serve as the study area for this project (Figure 1), however if model development is successful, this approach can be extrapolated to occupy a larger domain.

Observed wind data, including wind speed, direction, temperature, and pressure, will be obtained from NOAA's National Data Buoy Center, specifically from station NWPR1 (Figure 1) [13]. The observed data serves as the target value to assist in the validation of the model. The model makes predictions for U and V wind components at this location.

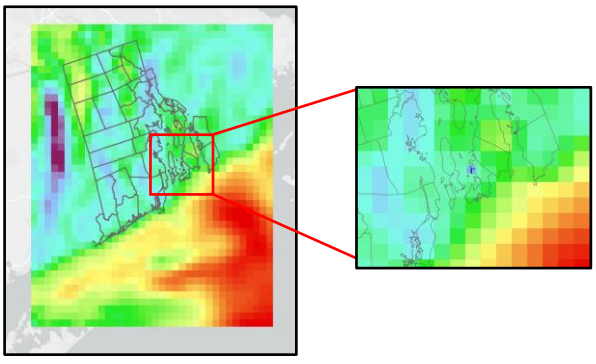

Figure 1: Data domain covering Rhode Island extent

B. Model Architecture

This study evaluates the performance of three model architectures for wind forecasting: CNN, LSTM, and a hybrid CNN-LSTM, as shown in Figure 2. The dataset includes U and V wind components, pressure, temperature, and humidity across a 48x36 GRIB grid, along with U and V target data from the NWPR1 weather station. Data is normalized to a -1 to 1 scale for consistent feature scaling and improved convergence and split into 60% training, 20% validation, and 20% testing for robust evaluation. Each model is trained over 25 epochs, with performance assessed through training and validation loss. The Adam optimizer is used with learning rates of 0.0001, 0.001, and 0.1. L2 regularization and dropout, with values of 0.0001, 0.001, 0.01, and 0.2, 0.3, 0.4 respectively, to facilitate a sensitivity analysis on model accuracy.

The CNN model processes spatial features of weather data through multi-layer convolutional and pooling operations. The architecture consists of three blocks. The first block applies 64 filters with a kernel size of 7x7. The second block applies 128 filters with a kernel size of 5x5. The third block applies 256 filters with a kernel size of 3x3. Each convolutional block includes ReLU activation, L2 regularization, batch normalization, max-pooling (2x2), and dropout. The extracted features are flattened and passed through fully connected layers with sizes 256, 128, and 64, also including L2 regularization and ReLU activation. The output layer is a fully connected layer with 2 filters and linear activation, suited for regression.

The LSTM model captures temporal dependencies using a sequence of past data points. The data is flattened and grouped into a 10-hour time window to enable learning sequential patterns. The architecture includes two LSTM layers. The first LSTM layer has 128 units and outputs sequences to the next layer. The second LSTM layer has 64 units and outputs a single feature vector for prediction. L2 regularization and dropout layers follow each LSTM layer to prevent overfitting. A fully connected layer with 64 units uses ReLU activation, and the output layer has 2 filters and linear activation.

The hybrid CNN-LSTM model combines the strengths of CNNs in capturing spatial features and LSTMs in learning temporal dependencies. This block consists of two convolutional layers. The first CNN layer applies 64 filters with a kernel size of 3 and ReLU activation, followed by max-pooling with a pool size of 2. The second CNN layer applies 128 filters with a kernel size of 3 and ReLU activation, followed by max-pooling with a pool size of 2. The spatial features extracted by the CNN layers are flattened and fed into the LSTM layers. The LSTM portion consists of two layers. The first LSTM layer has 128 units and outputs sequences to the next layer. The second LSTM layer has 64 units and outputs a single feature vector for prediction. To prevent overfitting, L2 regularization and dropout layers are applied throughout the architecture. The final output comes from a fully connected layer with 2 filters and linear activation.

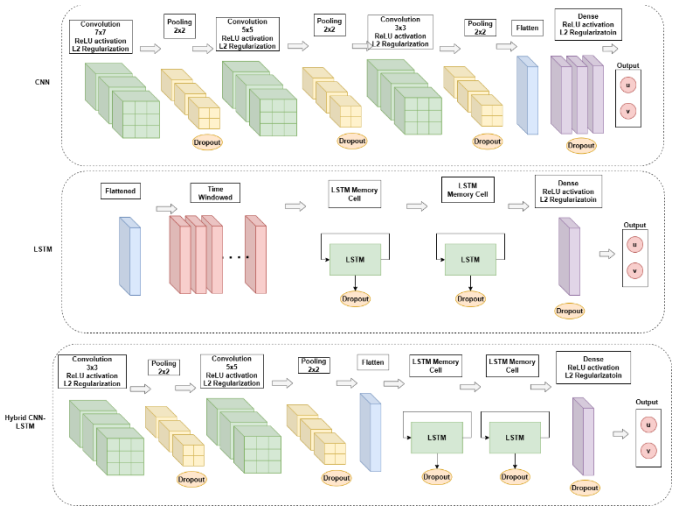

Figure 2: Model architecture for CNN, LSTM, and hybrid CNN-LSTM models

C. Model Evaluation

The dataset was divided into 60% training, 20% validation, and 20% test sets. Typically, datasets are split randomly, often incorporating cross-validation techniques to expose models to diverse temporal patterns and reduce the risk of overfitting. However, for Long Short-Term Memory (LSTM) models, preserving the temporal sequence is critical to maintaining the integrity of the time-dependent relationships in the data. Therefore, in this study, the sequential order of the data was preserved during the split.

The test set is designed to evaluate the model's ability to forecast wind speed and direction at the NWPR1 weather station. The model's performance was assessed using the Root Mean Squared Error (RMSE), which is calculated using Equation 1:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \quad (1)$$

Model accuracy was primarily evaluated against the HRRR model and relevant wind forecasting studies. The HRRR model is widely regarded as one of the most reliable tools for wind forecasting in this domain, yet its predictions are not without error. This inherent variability makes it a valuable benchmark for assessing the performance of the forecasting model. In addition to HRRR forecasts, accuracy was compared with results from studies employing similar techniques. For instance, one study forecasting surface wind speed in the Arctic utilized a CNN-LSTM model, achieving an RMSE of 0.4046 m/s [9]. While this serves as a useful reference, it is worth noting that many studies focus on wind turbine power generation as the target variable instead of wind speed and direction, which complicates direct comparison. To further contextualize performance, HRRR model accuracy was assessed by calculating its RMSE for observed wind speed and direction data at the NWPR1 weather station. This comparison allows the developed model to be evaluated not only in absolute terms but also relative to the current state-of-the-art in wind forecasting.

A significant challenge in benchmarking this study's model arises from the differences in target variables and experimental setups across studies. Many wind forecasting models are optimized for predicting wind turbine power output, which, while indirectly related to wind speed and direction, introduces additional variables and assumptions that are not relevant to this study's goals. Consequently, while direct RMSE comparisons can provide insights, contextual interpretation of these metrics is crucial.

D. Software Requirements

The Python package Herbie and the GRIB compiler wgrib2 were used to download and process the weather data [10, 12]. The Unity high-performance computing cluster was used during this process [13]. Once downloaded the xarray python package was used to read and extract the weather parameters [14]. The model architectures were developed using the Python packages sci-kit-learn and TensorFlow [15, 16]. The Conda environment manager was used to manage package dependencies.

IV. Results

The evaluation of the CNN, LSTM, and hybrid CNN-LSTM models provided an understanding of their performance and suitability for short-term wind forecasting. Each architecture was assessed for training and validation loss, as well as RMSE for both the U and V wind components. The results highlighted the advantages and limitations of each model.

A. Model Comparisons

Among the three architectures, the CNN model demonstrated superior performance in terms of computational efficiency and overall accuracy. Using a learning rate of 0.0001, a dropout rate of 0.3, and L2 regularization at 0.01, the CNN model achieved a training loss of 0.05 and a validation loss of 0.06. The model also achieved the lowest RMSE values for the U and V components, 0.21 m/s and 0.19 m/s, respectively, when compared to the benchmark dataset. These training and validation losses observed during model development are shown in Figure 3.

The best results from the LSTM model also used a learning rate of 0.0001, a dropout rate of 0.3, and L2 regularization at 0.01. This LSTM model exhibited moderate performance, achieving RMSE values of 0.27 m/s for U and 0.20 m/s for V under its best configuration. The training and validation loss for this LSTM model were 0.08 and 0.09, respectively. Although it effectively captured temporal dependencies, the additional computational demand and higher validation loss indicated that its advantages were not sufficient in this specific application.

The best results from the hybrid CNN-LSTM model used a learning rate of 0.0001, a dropout rate of 0.4, and L2 regularization at 0.01. This CNN-LSTM model, which combined spatial feature extraction and temporal sequence learning, achieved RMSE values of 0.23 m/s for U and 0.193

m/s for V. The training and validation loss was 0.05 and 0.06, respectively. Despite the hybrid model's ability to integrate both spatial and temporal insights, it did not outperform the standalone CNN model in this context. Furthermore, the hybrid model required significantly more computational resources, diminishing its practicality for this task.

Overall, the results suggest that the CNN model is best suited for short-term wind forecasting in coastal regions, balancing accuracy and computational efficiency. While the hybrid and LSTM models demonstrated marginal improvements in specific scenarios, their resource intensity and comparable or slightly lower accuracy make them less optimal for the given objectives.

Figure 3: Training and validation loss for best CNN model

B. *Sensitivity Analysis*

A thorough sensitivity analysis revealed the impact of learning rate, dropout, and L2 regularization on model performance. Across all architectures, learning rate emerged as the most sensitive parameter, with lower values generally resulting in better model convergence. Interestingly, the correlation between learning rate and RMSE values varied across the U and V components, suggesting nuanced interactions between the parameter settings and the specific wind dynamics being modeled. While dropout rates and L2 regularization also influenced model performance, their effects were less pronounced and more variable across architectures. An overview of the parameter sensitivity for the chosen CNN model is shown in Figure 4.

To evaluate the ranking of sensitivity of each parameter, a correlation matrix was constructed to quantify the linear relationship between the input parameters and the evaluation metrics, RMSE values for the U and V components on the test data. Correlation coefficients close to 1 indicate a strong positive correlation, while coefficients close to -1 indicate a strong negative correlation. Values near 0 suggest little linear relationship.

For instance, the CNN model exhibited stronger negative correlations for learning rate and L2 regularization, highlighting their role in mitigating overfitting. In the CNN model, the parameter rankings for RMSE U and RMSE V were L2 regularization (-0.17) > learning rate (-0.17) > dropout rate (0.02) and learning rate (-0.26) > L2 regularization (-0.18) > dropout rate (0.14), respectively. The LSTM model showed a different trend, with rankings for RMSE U and RMSE V being L2 regularization (0.41) > dropout rate (0.38) > learning rate (-0.32) and learning rate (0.43) > L2 regularization (0.27) > dropout rate (0.04), respectively. The hybrid CNN-LSTM model demonstrated yet another pattern, with dropout rate (0.33) > learning rate (-0.19) > L2 regularization (-0.03) for RMSE U, and learning rate (0.51) > dropout rate (0.28) > L2 regularization (0.26) for RMSE V.

These findings reinforce the importance of tailored parameter tuning to achieve optimal results for each architecture. Despite the variability in parameter sensitivity rankings, learning rate consistently played a pivotal role across all models, underscoring its critical influence on model convergence and accuracy.

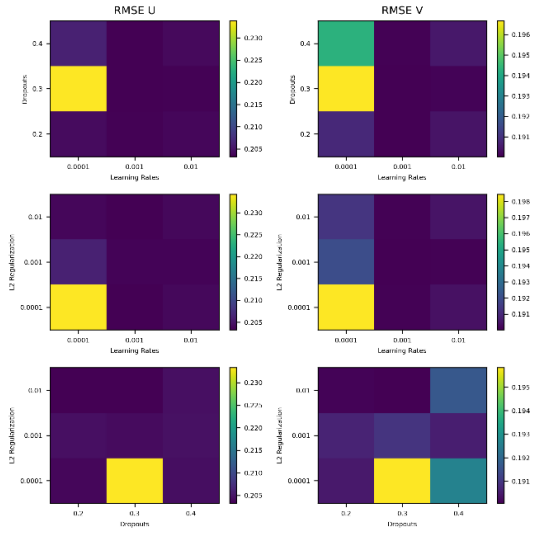

Figure 4: Sensitivity of input parameters

V. Conclusions

This study successfully developed and evaluated a hybrid CNN-LSTM model alongside standalone CNN and LSTM models for short-term wind forecasting in coastal regions. While all three models demonstrated the capability to predict wind speed and direction with reasonable accuracy, the CNN model emerged as the most effective architecture for this specific task. With RMSE values of 0.21 m/s for U and 0.19 m/s for V, the CNN model surpassed the benchmark HRRR data by 8.7% and 20.8% for U and V components, respectively. These improvements highlight the potential of deep learning approaches to enhance forecasting accuracy over conventional methods.

Interestingly, the LSTM model, despite its suitability for capturing temporal dependencies, did not provide significant advantages in this application. This outcome suggests that the spatial features extracted by the CNN model were more critical for short-term wind forecasting in the studied coastal environment. The hybrid CNN-LSTM model, while theoretically advantageous, failed to deliver a noticeable improvement in performance compared to the CNN model

alone. The additional computational overhead further reduced its practicality, underscoring the need for efficient architectures in resource-intensive applications.

The study also revealed key insights into parameter sensitivity, emphasizing the critical role of learning rate in model performance. These findings can guide future efforts to optimize deep-learning models for weather forecasting and other time-sensitive applications.

Building on the findings of this study, several areas for future research can be explored. Expanding the dataset to include more diverse meteorological conditions and geographic regions could improve model generalization. Additionally, experimenting with alternative architectures, such as support vector machines or transformer-based models, may uncover further enhancements. Incorporating additional weather forecast models, such as ECMWF or NBM, and blending their predictions with the developed model could also improve accuracy and robustness.

Finally, exploring the influence of coastal and topographic features on wind dynamics could provide valuable insights for refining model inputs. By addressing these areas, future studies can continue to advance the field of wind forecasting, enabling more reliable predictions and broader applications across industries.

VI. References

[1] Benjamin, S. G.; Weygandt, S. S.; Brown, J. M.; Hu, M.; Alexander, C. R.; Smirnova, T. G.; Olson, J. B.; James, E. P.; Dowell, D. C.; Grell, G. A. A North American hourly assimilation and model forecast cycle: The Rapid Refresh. Mon. Weather Rev. 2016, 144, 1669–1694.

[2] Li, Q.; Wang, G.; Wu, X.; Gao, Z.; Dan, B. Arctic short-term wind speed forecasting based on CNN-LSTM model with CEEMDAN. Energy 2024, 299, 131448.

[3] Liu, M.; Cao, Z.; Zhang, J.; Wang, L.; Huang, C.; Luo, X. Short-term wind speed forecasting based on the Jaya-SVM model. International Journal of Electrical Power & Energy Systems 2020, 121, 106056.

[4] Wang, A.; Xu, L.; Li, Y.; Xing, J.; Chen, X.; Liu, K.; Liang, Y.; Zhou, Z. Random-forest based adjusting method for wind forecast of WRF model. Comput. Geosci. 2021, 155, 104842.

[5] Chen, Y.; Wang, Y.; Dong, Z.; Su, J.; Han, Z.; Zhou, D.; Zhao, Y.; Bao, Y. 2-D regional short-term wind speed forecast based on CNN-LSTM deep learning model. Energy Conversion and Management 2021, 244, 114451.

[6] Zhu, X.; Liu, R.; Chen, Y.; Gao, X.; Wang, Y.; Xu, Z. Wind speed behaviors feather analysis and its utilization on wind speed prediction using 3D-CNN. Energy 2021, 236, 121523.

[7] Xu, X.; Wei, Y. An ultra-short-term wind speed prediction model using LSTM and CNN. Multimedia Tools Appl 2022, 81, 10819–10837.

[8] Wang, Y.; Zou, R.; Liu, F.; Zhang, L.; Liu, Q. A review of wind speed and wind power forecasting with deep neural networks. Appl. Energy 2021, 304, 117766.

[9] Li, Q.; Wang, G.; Wu, X.; Gao, Z.; Dan, B. Arctic short-term wind speed forecasting based on CNN-LSTM model with CEEMDAN. Energy 2024, 299, 131448.

[10] J. Hobson, “Herbie: High-Resolution Rapid Refresh (HRRR) Weather Model Data Retrieval,” Python, 2023. [Online]. Available: https://github.com/blaylockbk/Herbie

[11] "Unity High Performance Computing Cluster," University of Rhode Island, Kingston, RI, 2023. [Online]. Available: https://unity.uri.edu

[12] W. Ebisuzaki, "wgrib2: Utility for Grib2 Files," National Oceanic and Atmospheric Administration (NOAA), 2015. [Online]. Available: https://www.cpc.ncep.noaa.gov/products/wesley/wgrib2

[13] NOAA National Data Buoy Center, "Station NWPR1 - Newport, RI," National Data Buoy Center, National Oceanic and Atmospheric Administration (NOAA). [Online]. Available: https://www.ndbc.noaa.gov/station_page.php?station=nwpr1

[14] J. Walker, “pygrib: Python Interface to GRIB API,” Python, 2023. [Online]. Available: https://jswhit.github.io/pygrib/

[15] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/

[16] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011

[17] Sun, Zhuo, et al. "Machine Learning-Based Temperature and Wind Forecasts in the Zhangjiakou Competition Zone during the Beijing 2022 Winter Olympic Games." Journal of Meteorological Research 38.4 (2024): 664-679.