Harihara Nalamotu

Testing Different Kinds of Neural Networks to Analyze Patterns in Wildfires Across India

JES Journal of Electrical Systems

Abstract: **-** In recent years, due to climate change and global warming, the number of wildfires has dramatically increased. This increase in wildfires has led to property damage, injury and loss of life across the world. This study aims to develop a Neural Network capable of modeling the risk of wildfires across India using weather data. This would warn firefighters before a fire starts and would lower response time, hence minimizing damage. 4 Neural Networks were tested, a simple Convolutional Neural Network, as control, a Convolutional AutoEncoder and 3 types of Recurrent Neural Networks with Convolutional layers, to capture time complexity. The Neural Networks utilized Convolutional layers to capture location complexity. 4 models achieved acceptable levels of accuracy, ranging from a Mean Squared Error Loss of under 1 to nearly 2.3.

Keywords: Neural Networks (NNs), Convolutional Neural Networks (CNNs), Convolutional AutoEncoders (CAEs), Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), Long Short-Term Memory (LSTMs).

I. INTRODUCTION

Wildfires today are significantly more destructive than the wildfires of 20 years ago [1]. Every year, the annual area affected by wildfires increases by 5.4% [2]. Just in the past 5 years, the world has seen the California wildfires of 2019, the Australian bushfires of 2020 and the Canadian wildfires of 2023 combined burn 23.2 million hectares of land and kill tens of people [3]-[5]. These wildfires also have catastrophic effects on animal life in the area [4].

A predictive system that serves to warn firefighters of a risk of wildfires could serve to reduce response time, combating fires before they can burn significant area and harm the local fauna. It may also allow firefighters to be prepared for fires and lessen risks to their lives. A predictive system may also reveal trends in wildfire locations, helping effectively organize firefighters to rapidly respond to fires.

Neural Networks (NNs) were chosen to build this system due to their ability to capture non-linear relationships between data [6]. Convolutional layers were used to capture any location-based dependencies in predictions. Recurrent Neural Networks (RNNs) were tested to gauge how well they captured the time complexity of increasing wildfires mentioned. These models are capable of accurate time-series analysis, allowing the inclusion of any temporal trends in the model.

While there has been significant research conducted into the field of wildfire predictions [10]-[13], most of this work has been focused on algorithms modeling the spread of wildfires or using sensor and satellite data to make predictions. The systems proposed in this study are far cheaper, more practical and easier to implement, due to their only requirements being simple weather data. This study is also specific to India, and the models developed are specifically trained on Indian weather conditions.

II. METHODOLOGY

1. Data Collection

The wildfire data was collected using the NASA Fire Information for Resource Management System (FIRMS) API [8]. The dataset spanned 379 days, from October 1st, 2022, to October 14th, 2023, and contained nearly 1.5 million instances of wildfires across India.

The weather data, using VisualCrossing's Weather API [9], was collected for the same time range, but each data point represented a grid box of size 0.25 degrees East by 0.25 degrees North. The size of the grid was 121 by 121, leading to nearly 5.5 million rows of weather data.

The data was organized such that each grid box at each date had an individual "fire_count" variable, which tracked the number of wildfire instances within that grid box, as well as the temperature, humidity, windspeed and wind direction of the box. The "fire_count" variable was calculated individually using the wildfire instance dataset.

2. Models

Firstly, 2 different Convolutional Neural Network (CNN) architectures were trialed: conventional CNNs and Convolutional AutoEncoders (CAEs). The power of CNNs lays in feature extraction: they can use spatial data to extract location-based features to make inferences. For example, for the purpose of this study, CNNs are used to infer regional dependencies between wildfires. Next, 3 RNN architectures were trialed: simple RNNs, Gated Recurrent Units (GRUs) and Long Short-Term Memory Models (LSTMs). RNNs can capture temporal complexity across data, making them ideal for time-series analysis and other forms of modeling through time.

2.1. CNNs

CNNs work by taking an input as a matrix, generally a 3-dimensional array, representing width, height and channels. This is called the input layer. For the purpose of this study, the width and height will represent the latitudes and longitudes of the data, and the channels will represent the weather data. Then Convolutional layers are applied, which function by applying a set of filters over the layer, each of which generate a feature map for the area over which the filter is applied to. The higher the filter count, the smaller and more intricate the feature maps are. Higher filter counts can allow the model to find more complex patterns, however they can also lead to overfitting. CNNs were chosen for this study due to their ability to comprehend spatial relationships, for example, they may identify that there is a higher risk of wildfires if there are fires in the adjacent grid box.

2.1.1. Input Layer

The model has an input shape of (121, 121, 5). Representing a grid of size 121 x 121, where each box is 0.25 degrees East by 0.25 degrees North. Each grid box has 5 weather data points: temperature, humidity, windspeed, wind direction and fire count.

2.1.2. Convolutional Layers

The model has 4 Convolutional layers, with filter counts of 32, 64, 64 and 128 respectively. The activation function for these layers is a ReLU activation function, to introduce non-linearity and allow the model to capture more complex patterns.

2.1.3. Max Pooling Layers

There are 3 Max Pooling layers after the first 3 Convolutional layers to reduce dimensionality while retaining prominent features. This helps prevent overfitting.

2.1.4. Global Average Pooling Layer

There is 1 Global Average Pooling layer after the final Convolutional layer to reduce the data to a single vector which can then be processed by the fully connected layers.

2.1.5. Dense Layers

There are 2 Dense layers in the model. They are at the very end. The first one is meant to combine the features extracted by previous layers (as they are fully connected). The second Dense layer is an output layer.

2.1.6. Training

The model was trained for 100 epochs on a batch size of 128.

2.2. CAEs

CAEs work by taking a similar input as a matrix. However, they, while applying Convolutional layers, reduce the dimensionality of the data, allowing for the removal of noise and unnecessary data. This allows the model to capture the most important features of the data. The data is then reconstructed (the dimensionality is "added back") to provide an output. During this process, however, important information may be mistaken for noise and eliminated. CAEs were chosen for their strengths in robust feature identification due to their functionality of lowering dimensions. For a dataset with 1.5 million values, this is very important.

2.2.1 Encoder Function

The encoder function takes inputs and lowers the dimensionality of data to extract only the important features.

2.2.1.1. Input Layer

The encoder has an input shape of (121, 121, 5). Representing a grid of size 121 x 121, where each box is 0.25 degrees East by 0.25 degrees North. Each grid box has 5 weather data points: temperature, humidity, windspeed, wind direction and fire count. This is identical to the CNN.

2.2.1.2. Convolutional Layers

The encoder has 3 Convolutional layers, with filter counts of 32, 64 and 128 respectively. Just like the CNN, all of these layers use ReLU activation functions to introduce non-linearity and allow them to capture more complex patterns.

2.2.1.3. Max Pooling Layers

The Max Pooling layers in the encoder lower the dimensionality of the data after each Convolutional layer. The final Max Pooling layer provides an output with dimensions of (1, 16, 16).

2.2.2. Decoder Function

The decoder function takes the lowered-dimension outputs from the encoder function and applies Convolutional layers while also increasing the dimensionality back to the original input shape.

2.2.2.1. Convolutional Layers

The decoder function has 5 Convolutional layers, with decreasing filter counts of 128, 64, 32, 32 and 1, respectively. The final Convolutional layer serves to set the output shape as near as possible to the required shape. They do not use ReLU activation functions

2.2.2.2. Up Sampling Layers

These layers are used after the first 3 Convolutional layers to increase the dimensionality of the output back to the required shape.

2.2.2.3. *Cropping Layer*

The Cropping layer crops the output shape to the required (121, 121, 1).

2.2.3. Training

The model was trained for 100 epochs on a batch size of 128.

2.3. RNNs

RNNs are similar to CNNs, but rather than capturing spatial patterns, they capture temporal patterns. However, rather than using filters like CNNs, RNNs feed the output of the previous input into the model to make the current prediction. This allows it to keep a memory of temporal data. However, because only the output from the previous input is fed in, RNNs have poor comprehension of analyzing long-term dependencies. RNNs were chosen because of their proficiency at capturing short-term dependencies. The accuracy of the RNN could show the length of time over which wildfires are affected.

2.3.1. Model Architecture

The model is an RNN, constructed using the Keras inbuilt constructor for RNNs “Sequential”.

2.3.2. Input Layer

The model has an input shape of (121, 121, 5). Representing a grid of size 121 x 121, where each box is 0.25 degrees East by 0.25 degrees North. Each grid box has 5 weather data points: temperature, humidity, windspeed, wind direction and fire count. This is identical to both the CNN and CAE.

2.3.3. Convolutional Layers

The model has 4 Convolutional layers with filter counts of 32, 64, 64 and 128 respectively. It also uses ReLU activation functions to introduce non-linearity. It is identical to the Convolutional layers used in the CNN.

2.3.4. Max Pooling Layers

There are 3 Max Pooling layers after the first 3 Convolutional layers to reduce dimensionality while retaining prominent features. This helps prevent overfitting. This is also identical to the CNN.

2.3.5. Global Average Pooling Layer

There is 1 Global Average Pooling layer after the final Convolutional layer in order to reduce the data to a single vector which can then be processed by the fully connected layers, just like the CNN.

2.3.6. Dense Layers

There are 2 Dense layers in the model. They are at the very end. The first one is meant to combine the features extracted by previous layers (as they are fully connected). The second Dense layer is an output layer. Again, identical to the CNN.

2.3.7. Training

The model was trained for 100 epochs on a batch size of 128.

2.4. GRUs

Gated Recurrent Units, on the other hand, use "gates", which allow them to decide which information needs to be carried forward into future layers, and which can be discarded. The Update Gate decides how much past information must be carried forward, and the Reset Gate decides how much past information must be forgotten. These gates allow them to hold longer-term memory, allowing them to make better long-term temporal inferences. GRUs were chosen for a similar reason to the RNNs. GRUs are very good at capturing longer-term temporal complexity. Analyzing the accuracy of a GRU model will show how dependent wildfire predictions are on longer-term trends.

2.4.1. Model Architecture

Similar to the RNN, this model also uses the inbuilt Keras RNN constructor "Sequential".

2.4.2. Input Layer

This model also has an input shape of (121, 121, 5). Representing a grid of size 121 x 121, where each box is 0.25 degrees East by 0.25 degrees North. Each grid box has 5 weather data points: temperature, humidity, windspeed, wind direction and fire count. This is identical to all the other models.

2.4.3. Convolutional Layers

It is identical to both the RNN and CNN. There are 4 Convolutional layers with filter counts of 32, 64, 64 and 128 respectively. It also uses ReLU activation functions to introduce non-linearity. It is identical to the Convolutional layers used in the CNN.

2.4.4. Max Pooling Layers

Once more, it is identical to both the RNN and CNN. There are 3 Max Pooling layers after the first 3 Convolutional layers to reduce dimensionality while retaining prominent features. This helps prevent overfitting. This is also identical to the CNN.

2.4.5. Global Average Pooling Layer

There is 1 Global Average Pooling layer after the final Convolutional layer to reduce the data to a single vector which can then be processed by the fully connected layers, just like the CNN and RNN.

2.4.6. Reshaping Layers

The model has a Reshaping layer, which reshapes the vector received from the Global Average Pooling layer to a 2D matrix to be processed by the following layers.

2.4.7. GRU Layers

The model has 2 GRU layers, both with neuron counts of 128. These GRU layers allow the model to explore time complexity throughout the input data. They discard irrelevant temporal information and add important information to the "hidden state", which holds important data to be provided as an input for the next item.

2.4.8. Dense Layers

There are 2 Dense layers in the model. They are at the very end. The first one is meant to combine the features extracted by previous layers (as they are fully connected). The second Dense layer is an output layer. Again, identically to the CNN and RNN.

2.4.9. Training

The model was trained for 100 epochs on a batch size of 128.

2.5. LSTMs

LSTMs are another kind of RNN which, similarly to GRUs, use gates to maintain memory. However, LSTMs have an additional gate compared to GRUs: Input, Output and Forget. The Forget gate, similarly to the Reset gate in a GRU, decides what information is discarded. The Input gate, similar to the Update gate in a GRU, decides what information is added to the "cell state", the long-term memory of the model. The Output gate control the "hidden state", which the LSTM uses for shorter-term memory and output generation. LSTMs were chosen for this study for their ability to combine the strengths of both GRUs and RNNs. It was believed that an LSTM would provide the most accurate results.

2.5.1. Model Architecture

Similar to the RNN and GRU, this model also uses the inbuilt Keras RNN constructor "Sequential".

2.5.2. Convolutional LSTM Layers (ConvLSTM)

The model utilizes 4 Convolutional LSTM layers, all with filter counts of 64. They all also use ReLU activation functions to add non-linearity and allow the model to capture complex relationships.

2.5.3. Time Distributed Max Pooling Layers

Time Distributed layers allow the model to apply the same Max Pooling layers individually to each instance in the batch. The Max Pooling layers decrease the dimensionality of the data to capture only the relevant features.

2.5.4. Flatten Layer

The flatten layer reshapes the features into a 1-dimensional vector to be processed by the upcoming fully connected layers.

2.5.5. Dense Layers

There are 2 Dense layers in the model. They are at the very end. The first one is meant to combine the features extracted by previous layers (as they are fully connected). The second Dense layer is an output layer. Identically to the CNN, RNN and GRU.

2.5.6. Training

The model was trained for 50 epochs with a batch size of 128. This model was only trained for 50 epochs due to computational resource limitations. Training times for LSTMs are significantly higher than for any of the other models developed.

III. FINDINGS

Model Name	Non-recurrent CNN	Non-recurrent CAE	Recurrent CNN	CNN-GRU	ConvLSTM
Mean Squared Error (MSE) Loss	2.0315	3.8197	1.9821	2.2872	0.8734

Figure 1. Mean Squared Error (MSE) Loss for Each Model

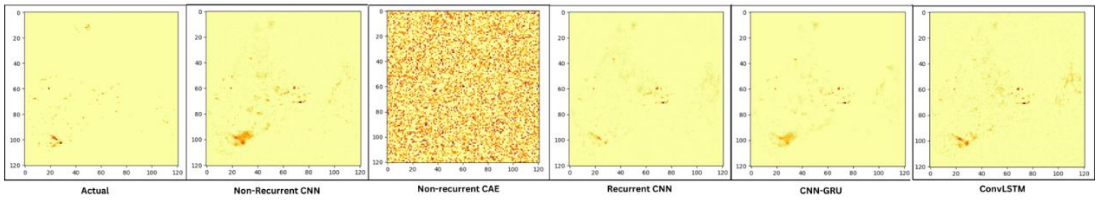

Figure 2. Comparison between Actual and Modeled Data

The graphs follow the pattern of MSE Loss, where the ConvLSTM appears most accurate, followed closely by the Simple Recurrent CNN, followed by the Non-Recurrent CNN and the CNN-GRU and finally the non-Recurrent CAE, which is nowhere near the actual results. Hence, for the rest of the study, the CAE will be ignored.

There are certain "patches" of high wildfire risk that are common across all the maps, in the Southwest corner. The ConvLSTM most accurately captures this. The Recurrent CNN decreases the fire risk somewhat, as evidenced by a lower quantity of bright red in the image. The Non-recurrent CNN increases the size of the patch and the CNN-GRU both increases the size and lowers the risk value, as evidenced by the lighter, more yellow color. All 4 models add additional "texture" to the maps. Across the entire map, they predict areas of slight wildfire risk that are not present in the Actual map. However, these are minuscule changes that do not affect the usability of the models in any real way.

IV. CONCLUSIONS

Wildfires pose a significant risk to human life and local fauna. There are no implemented wildfire prediction systems today. This causes ill-prepared firefighters, increased wildfire spread and higher risk to life. A predictive system could save time, natural resources and the environment, while also protecting firefighters.

This study has shown the efficacy of Neural Networks in the prediction of wildfires. It has shown that, among RNNs, LSTMs are most effective at predicting wildfires, while GRUs are least effective. This shows that, over the period the data was recorded for, long-term changes are not very significant. This is evidenced by the underperforming GRU model. The simple RNN outperformed the GRU, despite its poor comprehension of long-term dependencies. This shows that, more than long-term trends, short-term trends have more of an effect on

wildfires, alluding to the fact that wildfires are seasonal. However, LSTMs outperform the simple RNN by some margin, showing that there is at least some long-term dependency, despite it not being as prevalent as short-term seasonality.

Future improvements to this study include experimenting with more varieties of CNNs and RNNs, increasing the data range beyond 1 year and increasing the scope of the data from beyond the Indian peninsula.

V. REFERENCES

[1] Reuters. "Explained | How climate change drives heatwaves and wildfires in Europe." *The Hindu*, 18 Aug. 2023, www.thehindu.com/sci-tech/energy-and-environment/explained-how-climate-change-drives-heatwaves-and-wildfires-in-europe.

[2] MacCarthy, James. "The Latest Data Confirms: Forest Fires Are Getting Worse." *World Resources Institute*, www.wri.org/insights/global-trends-forest-fires.

[3] California Department of Forestry & Fire Protection. *2020 Fire Season Incident Archive | CAL FIRE*. www.fire.ca.gov/incidents/2020

[4] "In-depth: Australian bushfires | WWF-Australia | In-depth: Australian bushfires | WWF Australia." *WWF Australia*, wwf.org.au/what-we-do/australian-bushfires/in-depth-australian-bushfires.

[5] Canada, Natural Resources. *Canada's record-breaking wildfires in 2023: A fiery wake-up call*. 19 Aug. 2024, natural-resources.canada.ca/simply-science/canadas-record-breaking-wildfires-2023-fiery-wake-call/25303.

4 benefits of using artificial neural nets | Artificial Intelligence |. 20 Mar. 2017, www.allerin.com/blog/4-benefits-of-using-artificial-neural-nets.

[6] *4 benefits of using artificial neural nets | Artificial Intelligence* |. 20 Mar. 2017, www.allerin.com/blog/4-benefits-of-using-artificial-neural-nets.

[7] Z Zhou, C Qiu, Y Zhang. "A comparative analysis of linear regression, neural networks and random forest regression for predicting air ozone employing soft sensor models." *Scientific Reports*, vol. 13, no. 1, Dec. 2023, https://doi.org/10.1038/s41598-023-49899-0.

[8] "NASA-FIRMS." *NASA-FIRMS*, firms.modaps.eosdis.nasa.gov/api/area.

[9] Corporation, Visual Crossing. *Free Weather API | Visual Crossing*. www.visualcrossing.com/weather-api.

[10] Singh, H., Ang, LM., Lewis, T. *et al.* Trending and emerging prospects of physics-based and ML-based wildfire spread models: a comprehensive review. *J. For. Res.* **35**, 135 (2024). https://doi.org/10.1007/s11676-024-01783-x

[11] Ghali R, Akhloufi MA. Deep Learning Approaches for Wildland Fires Using Satellite Remote Sensing Data: Detection, Mapping, and Prediction. *Fire*. 2023; 6(5):192. https://doi.org/10.3390/fire6050192

[12] Spizzirri, John. "Lab develops unprecedented long-term wildfire prediction model." *Phys.org*, 7 Oct. 2020, phys.org/news/2020-10-lab-unprecedented-long-term-wildfire.html.

[13] Vinoth Kumar Kolluru, Advaitha Naidu Chintakunta, Yudisthir Nuthakki, Sonika Koganti. "Advancements in Wildfire Prediction and Detection: A Systematic Review." *IJFMR*, www.ijfmr.com/research-paper.php?id=23785.