10.48047/jocaaa.2024.33.08.196

Intelligent Framework for Flash Flood Forecasting using Modified Convolutional NeuralNetwork

Akanksha Varghese
Research Scholar, School of data Science & Forecasting
Devi Ahilya Vishwavidyalaya
Indore, India
akanksha.varghese@gmail.com
Dr. Mayank Saxena
Pro VC, Sage University
Indore, India
mayank.saxena71@gmail.com

Dr. Vandit Hedau
Associate Professor, School of Data Science and Forecasting
Devi Ahilya Vishwavidyalaya
Indore, India
vanditfsp@gmail.com

***Abstract*—**Sophisticated prediction tools are desperately needed since flooding is currently a major issue and will only get worse due to climate change. Your hybrid solution, which incorporates Recurrent Convolutional Neural Networks (RCNN) for stream stage height prediction, is one innovative way to address this problem. Traditional models like LSTM do a terrible job of capturing spatial relationships when it comes to flood prediction, but they perform admirably when it comes to temporal sequence modelling. Combining LSTM with Convolutional Neural Networks (CNNs) allows you to build a hybrid model that utilizes the best aspects of both techniques. CNNs are required for multi-modal inputs like remote sensing and in-situ measurements due to their outstanding pattern identification ability in spatial data. It's impressive that you were able to reduce model error by 26%. This result offers more proof that your model, which accounts for the complex spatiotemporal dynamics of flash flood events, can predict elevated stream conditions more correctly. Prediction accuracy must rise in order to strengthen early warning systems and facilitate more timely and informed local decision-making. This tactic might be highly helpful in areas that are prone to flooding, as early warnings could greatly minimize harm to people and property. Your research might have a big influence on flood prediction technology, resulting in the development of more accurate and dependable systems globally.

***Keywords*-Bidirectional CNN; Convolutional Neural Network;, flash flooding, decision support systems.**

10.48047/jocaaa.2024.33.08.196

I. INTRODUCTION

Flooding is still one of the most common and devastating natural disasters, causing billions of dollars' worth of losses each year [1]. The complicated interactions among an area's hydrology, geography, and climate dictate when and how severe floods occur. Flash floods account for the bulk of flood-related fatalities in the United States. The rapid water level response that peaks within six hours of intense rainfall characterises these floods. Because of the anticipated rise in climate extremes, flood damage is expected to become more severe in the coming decades. Many urban planners are concerned about more intense flooding and believe that improved forecasting of the occurrence and location of floods is necessary [2,3]. Some of the fastest basin reaction times were seen in areas located inside small, significant catchments. One such example is Ellicott City, a historic mill town in Howard County, Maryland [4]. The two "once-in-a-1,000-year" storm events that occurred in Ellicott City in 2016 and 2018 resulted in significant destruction and several fatalities (Halverson, 2019). The economic cost of response and recovery for the 2016 flood was estimated by [5] to be around $12 million, but the 2018 flood was estimated to have cost $10.5 million. In addition, there were additional costs related to lost revenue and a decrease in workforce in the area.

To reduce the probability of similar catastrophes in the future, Howard County developed many flood control plans. Sixty distinct scenarios were taken into consideration, encompassing the alteration of floodplains, the reclamation of buildings, the installation of new infrastructure for flood retention and diversion, and the enhancement of resilience measures. As part of this all-encompassing flood mitigation plan, a functioning flood forecast system for Ellicott City's historic downtown district has also been constructed. The ability to forecast flash floods is one of the most challenging parts of hydrometeorological research physical models are frequently used; these models attempt to characterize stream morphology, catchment features, and patterns of precipitation in order to represent the dynamics of rainfall-runoff. However, these techniques may require different parameterization for every stream, increasing their computing cost [6]. Research has demonstrated that the use of new datasets and techniques has improved the prediction of flash floods [7-9]. However, a novel strategy based on data-driven machine learning may be able to better simulate flood conditions at smaller spatial scales.

10.48047/jocaaa.2024.33.08.196

The general term "machine learning" encompasses any method that seeks to describe the relationship between input and output data [10]. Machine learning techniques may identify significant non-linearities in a hydrological system even in the absence of knowledge about the precise geophysical processes at play [11]. Time series problems such as streamflow prediction and flood risk assessment have found useful tools in the form of specific models known as artificial neural networks (ANN) [12]. Long short-term memory (LSTM) networks are a subclass of artificial neural networks (ANNs) that have wide applications in rainfall-runoff modelling due to their ability to process and retain information from longer sequences of data. These kinds of frameworks, which extract features from raw data using multilayer neural networks, are referred to as "deep learning". [13]

Recent advances in the field of hydrological deep learning have been made possible by convolutional neural networks (CNNs). The ability of convolutional neural networks (CNNs), a relatively new field of study, to process pictures and maintain spatial dynamics sets them apart from long short-term memory (LSTMs); this trait may prove helpful in resolving the flash flood prediction problem [14]. Soil moisture "memory" in the days following a rain event is one example of a landscape component that might affect the likelihood of subsequent floods. Therefore, spatiotemporal data may be very helpful in flash flood prediction. Particularly in places like Ellicott City, operational usage of current and projected stream conditions is used to initiate emergency public alerts and evacuations. We offer a ConvLSTM model architecture that integrates spatial and temporal data with conventional linear regression in order to enhance flash flood forecasts in Ellicott City, Maryland. We begin with a simple Long Short-Term Memory (LSTM) to predict when the Hudson Branch stream gauge will reach higher stages. We next contrast this initial state with models derived from hydrometeorological data, which are capable of capturing the basin's distinct landscape response. We evaluate the prediction accuracy, high stream condition detection ability, and flood simulation performance of the two models in 2016 and 2018.

This section provides an overview of the study's topic and earlier attempts in the deep learning flood forecasting area. In Section 3, the formulations of the baseline and ConvLSTM models are explored, along with the details of data collection and processing. Sections 4 and 5 contain the results and comments, respectively. Before delving into the model's specific applications in real-world flood control in Howard County, we finally discuss the model's shortcomings and the need for more research. The research is concluded with a few closing remarks in Section 6.

10.48047/jocaaa.2024.33.08.196

II. RELATED WORK

A number of Hydrological scientists have been studying deep learning with neural networks for a long time [15]. Because they address the "vanishing gradient" issue that plagued previous recurrent neural networks (RNNs), LSTMs have seen extensive use. This issue occurs when the model training signal decreases in strength as it travels through longer sequences of data ([16]. This is accomplished by use of LSTM cell-specific features that regulate the data retention during training. A given input, x_t, the previous cell output, c_{t-1}, and the previous cell state, h_{t-1}, are used to update the model weights at a certain timestep, t. During training, the output gate (ot) is responsible for sending information about the current state to the next stage of the process, while the forget gate (ft) decides which previous states should be deleted. A few of the recent research that show how LSTM networks outperform other physically based hydrological models.[16] Convolutional neural networks are more appropriate for picture or video classification than LSTMs because of their ability to handle input hierarchically in array form [17]. A number of recent research have shown that CNNs may be used for mapping flood vulnerability and extent, detecting extreme weather, and estimating predicted precipitation. The hybrid method, ConvLSTM, was suggested as part of their deep learning demonstration for precipitation nowcasting [17]. Instead of using matrix multiplication for the input and state-to-state transitions, they used convolutional operators [18]. From soil moisture prediction to flood extent mapping, from precipitation nowcasting to other neural network frameworks, this hybrid approach consistently outperforms the competition.

Few studies have directly applied ConvLSTM models to streamflow or stage, despite the models' shown effectiveness in many hydro meteorological applications. A model for flood forecasting was created by [18] using daily rainfall data from nine different locations in Fiji to calculate an Index (IF). The ConvLSTM system. The model was shown to be most effective in predicting daily stream conditions. However, the authors did point out that the study might have used more input characteristics and that hourly measurements would have been beneficial. Monthly streamflow data and an El Niño-Southern Oscillation (ENSO) index are used as inputs in a flood forecasting model that is based on ConvLSTM, as presented by [19].. Here, 2-dimensional inputs were used for model training from 1-dimensional individual datasets. Neither study specifically tackles flash

10.48047/jocaaa.2024.33.08.196

flooding or uses spatially-distributed inputs, but both have improved operational flood forecast modelling.

III. ARCHITECTURAL FRAMEWORK OF RECURRENT CNN

The recurrent convolutional layer (RCL) is the main component of an RCN. RCL unit states change at discrete time intervals. An RCL's net input $z_{ijk}(t)$ at time step t for a unit at (i, j) on the k^{th} feature map is as follows: [20]

$$z_{i,m,n}(t) = \left(w_n^f\right)^T u^{(i.m)}(t) + \left(w_n^r\right)^T x^{(i.m)}(t-1) + b_k \quad (1)$$

Here, the feedforward $u^{(i.m)}(t)$ and recurrent input $x^{(i.m)}(t-1)$ in the equation are represented by the vectorised patches centred at (i,m) of the feature maps in the previous and current layers, respectively. The bias is represented by the variable b_k.

This unit's activity or condition depends on its net input given as

$$x_{i,m,n}(t) = g\left(f\left(z_{i,m,n}(t)\right)\right) \quad (2)$$

where f is the rectified linear activation function. Also,

$$f\left(z_{i,m,n}(t)\right) = \max\left(z_{i,m,n}(t), 0\right)$$

Also, the local response normalisation (LRN) function can be denoted as

$$g\left(f\left(z_{i,m,n}(t)\right)\right) = \frac{f_{i,m,n}(t)}{\left(1 + \frac{\alpha}{N} \sum_{k'=\max\left(0, k-N/2\right)}^{\min\left(K, k+N/2\right)} \left(f_{i.m.n'}\right)^2\right)^{\beta}}$$

where g is the local response normalisation (LRN) function and K is the number of feature maps. The parameters α and β regulate the normalisation amplitude. The lateral inhibition seen in the cortex, where several aspects vie for significant responses, is modelled by LRN. In our concept, LRN is employed to keep the states from blowing up. Equations (1) and (2) explain how the RCL behaves dynamically. This layer may be unfolded over T time steps, yielding a feed-forward subnetwork of depth T + 1. The feed-forward input doesn't change during an iteration, but the recurrent input does. Only the feedforward input is present when t = 0. There are several pathways in the subnetwork connecting the input layer to the output layer. The shortest path just passes via the feed-forward connection (length = 1), but the longest path passes through every unfolded recurrent link (length = T + 1). As the number of iterations rises, the effective RF of an RCL unit in the previous layer's feature maps grows. The effective RF of an RCL unit is square, with a side

10.48047/jocaaa.2024.33.08.196

length of $(L_{rec}-1)T + L_{feed}$, where L_{feed} and L_{rec} stand for the side lengths of the input and recurrent filters, respectively, if both the input and recurrent filters in equation (1) have square forms in each feature map. A stack of RCLs, sometimes interspersed with max pooling layers, makes up an RCNN.

The architecture employed in this work is shown in Figure 3. Max pooling comes after layer 1, which is the typical feed-forward convolutional layer without recurrent connections to save computation. Four RCLs are utilised on top of this, with a max pooling layer in the centre. There are only feed-forward connections between adjacent RCLs. There are two pooling operations: size 3 and stride 2. After a global max pooling layer that outputs the maximum across each feature map, the fourth RCL's output produces a feature vector that represents the picture. This model differs from the ones in [22], which employ completely linked layers, which use global average pooling. The feature vectors are finally classified to C categories using a softmax layer.

The backpropagation through time (BPTT) technique is used to minimise the cross-entropy loss function during training [23]. This is the same as applying the time-unfolded network to the normal BP method. The total of a shared weight's gradients across all time steps is its ultimate gradient. The model becomes an extremely deep feed-forward network with $4(T+1)+2$ parameterised layers if we unfold the recurrent connections over T time steps. T + 1 represents the depth of each RCL. However, there are several alternative pathways with various lengths; $4(T+1)+2$ is just the length of the longest path from the input layer to the output layer. Of them, the feedforward path that avoids all recurrent connections has the shortest length, measuring six.

10.48047/jocaaa.2024.33.08.196

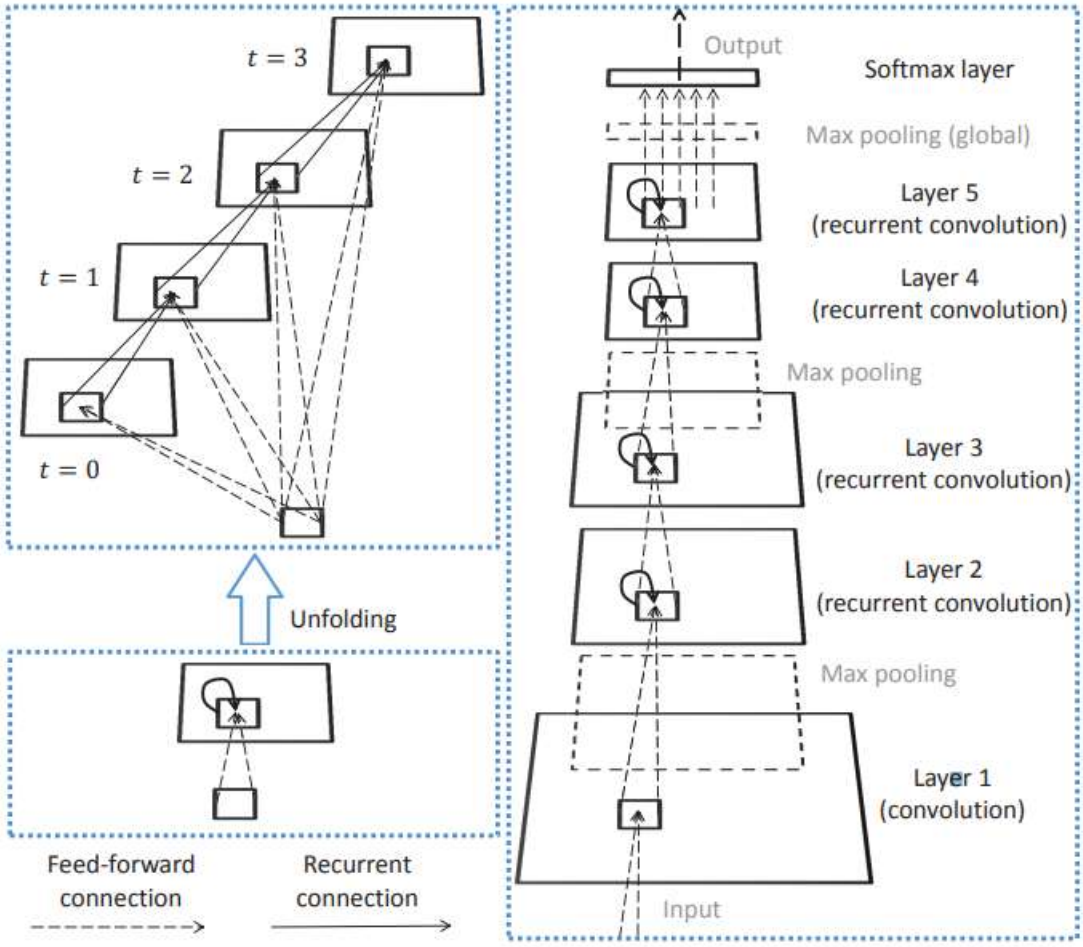

Fig.1. Recurrent CNN Framework [24]

Figure 1 displays the RCNN framework. Left: By unfolding an RCL for T = 3 time steps, a feed-forward subnetwork with a maximum depth of 4 and a minimum depth of 1 is created. At time t = 0, there is no computation performed other than feed-forward. The RCNN employed in this work consists of one convolutional layer, four RCLs, three max pooling layers, and one softmax layer.

IV. PROPOSED RCNN BASED FLASH FLOOD PREDICTION MODEL

In this section of the study, a framework for heart disease early detection is proposed. Numerous methods have been accepted for the prediction of cardiac conditions. In this study, we forecast cardiac illnesses using a convolutional neural network (CNN). Computer-based applications are necessary for data assessment in systems that primarily consist of computer-aided diagnostic techniques. The tedious, demanding, and stressful process of transferring information from a domain expert to a computer algorithm mostly relies on the judgement of the medical expert. CNN

10.48047/jocaaa.2024.33.08.196

was employed as a deep learning approach for prediction in order to effectively tackle this situation. The suggested methodology's architecture is depicted in Figure 3.

Model formulations that incorporate spatiotemporal data inputs are compared to the baseline LSTM's performance. The National Hydrography Datasets Plus (NHDPlus) High Resolution dataset from the U.S. Geological Survey (USGS) was used to establish the study's geographic domain (Moore et al., 2019).

An LSTM with a global attention mechanism that was implemented simply was used to build a baseline model. By illustrating global relationships between input and response variables, attention processes are employed to assist in finding pertinent information in lengthy data sequences [25] Stage height readings from earlier timesteps at the Hudson Branch and Patapsco Ellicott gauge stations serve as data inputs for the baseline model (Table 1). Stage height at time n+1 was predicted using a time series of stream observations with different input windows (1, 2, 3, 4, 6, and 8 hours). OneRain provided stream measurements spanning from January 2016 to October 2020, which were then resampled to match hourly indices for 42,384 samples (2016-01-01 00:00 to 2020-10-31 23:00). 70% of the data, which covered the period from January 2016 to May 2019, were used to train the baseline model. It should be mentioned that the floods of 2016 and 2018 occurred during this training time. When anticipating heightened flood conditions, a diagnostic test that divided the 2016 and 2018 floods into training and validation sets produced a 52% increase in RMSE. Models were verified using the last 30% of data, which covered the period from May 2019 to October 2020. Table 1 displays all of the configuration settings for the basic LSTM.

Table 1 Data Input for BaseLineLSTM and ConvLSTM

Model	Notation	Input	Source
Baseline	LSTM	*OneRain Observations* Stage height (Hudson Branch/Patapsco Ellicott)	howardcounty.onerain.com
ConvLSTM	A	*NEXRAD Mosaic* 8-bit Base Reflectivity	mesonet.agron.iastate.edu/docs/nexrad_mosaic
	B	*Noah LSM* Soil Moisture	disc.gsfc.nasa.gov/datasets/NLDAS_NOAH0125_H_002/summary?keywords=noahNoah
	C	*IMERG Final Precipitation L3* Precipitation Rate	disc.gsfc.nasa.gov/datasets/GPM_3IMERGHH_06/summary?keywords=imerg
	D	*KLWX Level-III NEXRAD* 1-hr Accumulated Precipitation	console.cloud.google.com/storage/browser/gcp-public-data-nexrad-l3 (public dataset accessible free with Google account authentication)

Table 2Configuration Details

10.48047/jocaaa.2024.33.08.196

Layer (type)	Output shape	Param #	Connected to
GAUGE (InputLayer)	[(None, 3, 2)]	0	
masking_(Masking)	(None, 3, 2)	0	GAUGE[0][0]
lstm_(LSTM)	(None, 3, 16)	1,216	masking[0][0]
last_hidden_state (Lambda)	(None, 16)	0	lstm[0][0]
attention_score_vec (Dense)	(None, 3, 16)	256	lstm[0][0]
attention_score (Dot)	(None, 3)	0	last_hidden_state[0][0] attention_score_vec[0][0]
attention_weight (Activation)	(None, 3)	0	attention_score[0][0]
context_vector (Dot)	(None, 16)	0	lstm[0][0] attention_weight[0][0]
attention_output (Concatenate)	(None, 32)	0	context_vector[0][0] last_hidden_state[0][0]
attention_vector (Dense)	(None, 8)	256	attention_output[0][0]
dropout_3 (Dropout)	(None, 8)	0	attention_vector[0][0]

Due to their moderate geographical resolution, minimum of hourly readings for temporal resolution, and suitability for rainfall-runoff modelling, four spatiotemporal datasets were chosen. The Integrated Multi-satellitE Retrievals for GPM (IMERG) technique was used to get remotely sensed precipitation data from the Global Precipitation Measurement (GPM) mission of the National Aeronautics and Space Administration (NASA). The Goddard Earth Sciences Data and Information Services Centre [25] provided the soil moisture data from the NLDAS Noah Land Surface Model (LSM). The Iowa Environmental Mesonet at Iowa State University provided the mosaicked base reflectivity from NOAA's Next Generation Radar (NEXRAD). The 1-hour cumulative rainfall data for NEXRAD Level III was obtained from public data sources on Google Cloud. The Python ARM Radar Toolkit was utilised to convert Level III radar data from the KLWX doppler station in Sterling, Virginia, from radial to grid format [26]. In order to match the hourly indices of the gauged response data, all datasets were temporally filtered to the nearest hour. While the KLWX data included many timesteps for which there was no accessible data, the NEXRAD Mosaic, Noah LSM, and GPM IMERG data are available at CONUS scales and at regular intervals. Throughout the training phase, masked arrays were used to analyse the timesteps for the missing KLWX data (n = 1,405). Then, using QGIS's "Nearest Neighbour" function, all datasets were spatially resampled to a horizontal resolution of 1 km in order to match them with the 36 by 48-kilometer research domain grid.

Here, before being concatenated with the baseline LSTM, all four hydrometeorological inputs are processed using the Keras ConvLSTM2D and Conv3D hidden layers with the Rectified Linear Unit (ReLU) activation function. The same variable input windows as in the baseline model were

10.48047/jocaaa.2024.33.08.196

used to train each variant. Google's open-source Keras package for Python with the TensorFlow backend was used to build the Baseline LSTM and ConvLSTM neural networks [27]. Using NVIDIA V100 GPUs, the NASA Centre for Climate Simulation (NCCS) machine learning cluster was used to train all of the models for 100 epochs. To allow for randomised initialisation circumstances, training and validation for each version were done five times, yielding 480 distinct model runs.

After collection, the data move on to the preparation stage. The method described in the data preparation section is used to fill in or eliminate the missing values. A number of prepossessing methods were used on the dataset in order to enhance it. The training and testing datasets were created by splitting the cleaned dataset into two smaller subsets. Using training data from the CNN algorithm, the model was trained using the training dataset. The results of the tests were applied. Figure 2 depicts the suggested approach for flash flood prediction, which consists of three primary stages. Preprocessing and feature extraction from the dataset constitute the first stage. The BiLSTM model's definition and training are the focus of the second phase, while the flash flood prediction framework is the focus of the third. Below is a detailed discussion of each of these phases.

10.48047/jocaaa.2024.33.08.196

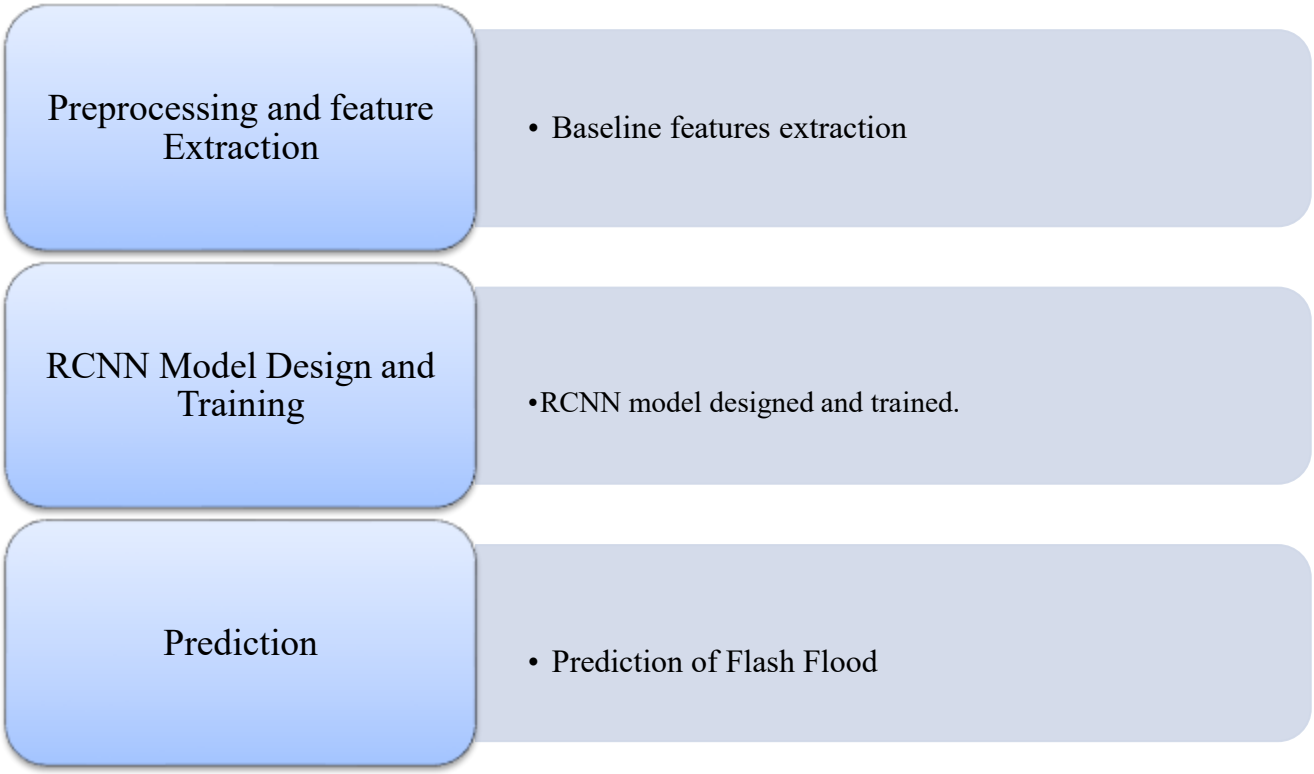

Fig.2 Proposed Framework

(a) Classification

The primary goal of research in proprietary protein engineering is to identify the protein sequences that work best to promote substrate exchange. Therefore, machine learning techniques are required to identify which protein sequences are more likely to exchange and which are less likely. We have so developed a two-tiered positive/negative categorisation system specifically for this scenario. This statistic assesses the categorisation performance of the prediction model based on the actual measured transaction. Training data were used to train the classifier that was developed using the recommended model. The model classified the data into two groups: "0" for no risk of flooding and "1" for flooding. The model generated a 91.71% prediction accuracy.

We consider not only the overall accuracy of each model, but also its ability to recognise elevated flood conditions using a predefined threshold. Based on the whole observational record from OneRain, we calculate a threshold stage height of 254.32 feet using the find_peaks function in Python's SciPy library and a statistical prominence of 0.75 (Figure 5). The "Significant" level for

10.48047/jocaaa.2024.33.08.196

the Hudson Branch site was established by the Howard County SWM, and it is around one foot lower than this height. Timesteps that have stage heights higher than this threshold are called "peak indices" (n = 164), and the model's independent objective is to evaluate the accuracy at these timesteps.

V. IMPLEMENTATION AND RESULTS

The objective of the experimental setup of the proposed model was to find and perceive the convolutional neural network's (CNN) performance using the dataset available at the UCI repository. When creating models to predict floods, it is important to take into account the full effects, anomalous changes, and spatiotemporal fluctuations of the parameters along with the underlying surface conditions. The advent of new complex flood processes with different spatiotemporal distributions will lead to variations in the parameters of hydrological models. This paper develops a unique approach to investigate the mechanism of flood production in a basin using Big Data and explainable AI. This methodology uses just the low-dimensional rainfall data to analyse the mechanisms that lead to floods. The main elements of this strategy are dynamic parameter migration, machine learning modelling, and pattern clustering analysis.

In the previous study, we discussed how to model the rainfall-runoff cycle by optimising TCN's network structure and hyperparameters. We evaluated the performance of the TCN model with four other models: LSTM, EIESM, TCN (without NDVI input), and TCN. We used the same experiment and ran simulations at the same forecast period of 1–12 hours as TCN. Table 2 displays the runoff forecasting evaluation index for each of the four models for different prediction periods (1–12 hours). The variations in the three indicators show how accurate the flood simulation was on the training and testing settings. In addition, we calculate the assessment metrics for every instance of flooding on the trial set to enhance the model's understandable representation of its performance.

After it had been prepared, the dataset was used to run future model runs. Convolutional Neural Network (CNN) training and deployment of the proposed model was carried out using Google Collaboratory (Google Colab). TensorFlow and Python were utilised to implement the entire model. CNN served as the model's main structural component. We also incorporated an eight-head multi-head self-attention mechanism. We employed CNN with the Nadamoptimisation approach

10.48047/jocaaa.2024.33.08.196

[51] for data classification. 0.001 was used as the initial learning rate. We employed an epoch count of 100 and a batch size of 32. After training the model successfully, it was able to obtain 90.12% accuracy on training data and 91.71% accuracy on validation data.

Table 3: Performance Metrix

Technique	Accuracy
LSTM	0.717
ANN,	0.759
EIESM	0.814
Proposed Work based on RCNN	0.894

EIESM is not entirely accurate, even though it expresses the flood process trend precisely. In addition, the predictive power of ANN and LSTM decreases significantly, underestimating flood peaks becomes more frequent, and simulated values diverge significantly from observed values. The large oscillations and odd outcomes of ANN may be due to its limited ability to learn from long-term data. Given that the actual peak flow is higher than the one anticipated by the LSTM, the flood warning system will be severely impacted. It is evident that TCNs perform better than other models in terms of properly predicting floods and simulating rainfall-runoff.

VI. CONCLUSION

This study provides an RCNN learning model for rainfall-runoff prediction by handling input and output sequences using a one-dimensional convolution operation. The key idea was to build each feed-forward convolutional neural network's convolutional layer with recurrent connections. The structure made it possible for units in the same layer to modulate the units, which enhanced the CNN's ability to identify statistical patterns in the surroundings of the object. Recurrent connections improved the depth of the original CNN by spreading weight among layers to keep the number of parameters constant. The TCN model, which generates the flow from the outflow component, is fed data from the rainfall stations in the basin, including rainfall, evaporation, and NDVI. The kernel size, filter size, and residual blocks are some of the RCNN hyperparameters that need to be considered throughout the modelling process. In case the combinations of hyperparameters are insufficient, the real value will not match the forecast results..

10.48047/jocaaa.2024.33.08.196

References

[1] M. M. A. Syeed, M. Farzana, I. Namir, I. Ishrar, M. H. Nushra, and T. Rahman, "Flood prediction using machine learning models," in *Proc. 2022 Int. Congr. Human-Computer Interact., Optimization and Robotic Appl. (HORA)*, Ankara, Turkey, Jun. 2022, pp. 9–11.

[2] H. Han, C. Choi, J. Jung, and H. S. Kim, "Deep learning with long short-term memory based sequence-to-sequence model for rainfall-runoff simulation," *Water*, vol. 13, no. 4, p. 437, 2021.

[3] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Transl. Vis. Sci. Technol.*, vol. 9, no. 14, p. 14, 2020.

[4] F. Kratzert, D. Klotz, M. Herrnegger, A. K. Sampson, S. Hochreiter, and G. S. Nearing, "Toward improved predictions in ungauged basins: exploiting the power of machine learning," *Water Resour. Res.*, vol. 55, pp. 11344–11354, 2019.

[5] S. Ha, D. Liu, and L. Mu, "Prediction of Yangtze River streamflow based on deep learning neural network with El Niño–Southern Oscillation," *Sci. Rep.*, vol. 11, pp. 1–23, 2021.

[6] S. A. Rahman and D. A. Adjeroh, "Deep learning using Convolutional LSTM estimates biological age from physical activity," *Sci. Rep.*, vol. 9, pp. 1–15, 2019.

[7] Y. Wang, Z. Fang, H. Hong, and L. Peng, "Flood susceptibility mapping using convolutional neural network frameworks," *J. Hydrol.*, vol. 582, p. 124482, 2020.

[8] C. Wobus et al., "Climate change impacts on flood risk and asset damages within mapped 100-year floodplains of the contiguous United States," *Nat. Hazards Earth Syst. Sci.*, vol. 17, pp. 2199–2211, 2017.

[9] N. I. Ulloa, S.-H. Yun, S.-H. Chiang, and R. Furuta, "Sentinel-1 spatiotemporal simulation using convolutional LSTM for flood mapping," *Remote Sens.*, vol. 14, p. 246, 2022.

[10] S. Abdullahi, M. Habaebi, and N. Malik, "Flood Disaster Warning System on the go," in *Proc. 7th Int. Conf. Comput. Commun. Eng.*, 2018, pp. 258–263.

[11] F. Liu, F. Xu, and S. Yang, "A Flood Forecasting Model based on Deep Learning Algorithm via Integrating Stacked Autoencoders with BP Neural Network," in *Proc. IEEE 3rd Int. Conf. Multimedia Big Data*, 2017, pp. 58–61.

10.48047/jocaaa.2024.33.08.196

[12] J. Noymanee, N. Nikitin, and A. Kalyuzhnaya, "Urban Pluvial Flood Forecasting using Open Data with Machine Learning Techniques in Pattani Basin," in *Proc. 6th Int. Young Sci. Conf. HPC and Simul.*, 2017, pp. 288–297.

[13] Y. Wu, Y. Ding, and J. Feng, "Sparse Bayesian Flood Forecasting Model Based on SMOTEBoost," in *Proc. Int. Conf. Internet Things (iThings) and IEEE Green Comput. Commun. (GreenCom) and IEEE Cyber, Phys. Soc. Comput. (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 279–284.

[14] S. Albawi, T. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. Eng. Technol.*, 2017, pp. 1–6.

[15] S. Marzukhi, M. Sidik, H. Nasir, Z. Zainol, and M. Ismail, "Flood Detection and Warning System (FLoWS)," in *Proc. 12th Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2018, pp. 1–4.

[16] N.-A. Maspo et al., "Evaluation of Machine Learning approach in flood prediction scenarios and its input parameters: A systematic review," in *IOP Conf. Ser.: Earth Environ. Sci.*, Bristol, UK: IOP Publishing, 2020.

[17] P. Mitra et al., "Flood forecasting using Internet of things and artificial neural networks," in *Proc. 2016 IEEE 7th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Vancouver, BC, Canada, Oct. 2016.

[18] J. Noymanee, N. O. Nikitin, and A. V. Kalyuzhnaya, "Urban pluvial flood forecasting using open data with machine learning techniques in Pattani basin," *Procedia Comput. Sci.*, vol. 119, pp. 288–297, 2017.

[19] G. Wang et al., "Application of a novel artificial neural network model in flood forecasting," *Environ. Monit. Assess.*, vol. 194, no. 125, 2022.

[20] S. Puttinaovarat and P. Horkaew, "Flood forecasting system based on integrated big and crowdsource data by using machine learning techniques," *IEEE Access*, vol. 8, pp. 5885–5905, 2020.

[21] M. S. Al Reshan et al., "A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing," *IEEE Access*, vol. 11, pp. 11390–11404, 2023.

[22] N. Islam et al., "Forecasting on COVID-19 Data Using ARIMAX Model," in *Data Sci. with Semantic Technol.*, Boca Raton, FL, USA: CRC Press, 2023, pp. 95–113.

10.48047/jocaaa.2024.33.08.196

[23] N. Islam et al., "Stock Prediction for ARGAAM Companies Dataset," *KIET J. Comput. Inf. Sci.*, vol. 6, pp. 1–13, 2023.

[24] J. Noymanee, N. Nikitin, and A. Kalyuzhnaya, "Urban Pluvial Flood Forecasting using Open Data with Machine Learning Techniques in Pattani Basin," in *Proc. 6th Int. Young Sci. Conf. HPC and Simul.*, 2017, pp. 288–297.

[25] K. Menon and L. Kala, "Video surveillance system for real-time flood detection and mobile app for flood alert," in *Proc. Int. Conf. Comput. Methodol. Commun.*, 2017, pp. 515–519.

[26] S. Han and P. Coulibaly, "Bayesian flood forecasting methods: A review," *J. Hydrol.*, vol. 551, pp. 340–351, 2017.

[27] W. Segretier, M. Collard, and M. Clergue, "Evolutionary predictive modelling for flash floods," in *Proc. IEEE Congr. Evolutionary Comput.*, 2013, pp. 844–851.