Flood Forecasting Model Using Federated Learning

Kopparthi Durga Devi
PG scholar, Department of MCA, CDNR collage, Bhimavaram, Andhra Pradesh.
K.Venkatesh
(Assistant Professor), Master of Computer Applications, DNR collage, Bhimavaram, Andhra Pradesh.

Abstract

The increasing sophistication of web attacks such as SQL injection, Cross-Site Scripting (XSS), and Distributed Denial of Service (DDoS) has elevated the need for intelligent and adaptive detection mechanisms. Traditional rule-based systems are no longer sufficient due to their inability to detect novel and evolving threats. This research aims to investigate and compare the effectiveness of different Machine Learning (ML) and Deep Learning (DL) algorithms in detecting web attacks. Algorithms such as Support Vector Machines (SVM), Decision Trees, Random Forests, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) are explored and benchmarked. The study uses widely recognized datasets like CICIDS2017 and UNSW-NB15 to evaluate the models under consistent conditions. Metrics such as accuracy, precision, recall, and F1-score are used to gauge the performance of each algorithm. The results indicate significant differences in how well each model detects certain types of attacks, and suggest the potential of ensemble and hybrid models for real-world web security systems. The goal is to contribute toward more robust and scalable intrusion detection solutions.

Introduction

The surge in web applications and online services has resulted in increased vulnerability to cyberattacks, which can compromise sensitive data and disrupt services. Web attacks are not only frequent but also dynamic in nature, constantly adapting to bypass security mechanisms.

Traditional intrusion detection systems (IDS) rely on static signatures, which means they cannot detect zero-day attacks or obfuscated threats. This has prompted researchers to explore data-driven approaches that can generalize from existing threats to predict unseen ones.

Machine Learning and Deep Learning methods offer powerful tools for automated pattern recognition and anomaly detection. These methods can learn complex relationships from traffic logs and system behavior without requiring manual intervention.

The primary objective of this research is to analyze and benchmark various ML and DL techniques to determine which algorithms are best suited for different web attack scenarios. The analysis will aid cybersecurity professionals in selecting and deploying the most appropriate solutions

Literature Survey

1. Many studies have highlighted the potential of ML in intrusion detection. SVM and Random Forests have been shown to offer high classification accuracy for structured network traffic data, with relatively low computational overhead.

2. DL models such as CNNs and LSTMs have gained popularity due to their ability to learn hierarchical representations from raw data. These models have shown exceptional performance in tasks requiring sequential or temporal data analysis, such as identifying attack patterns over time.

3. Some researchers have developed hybrid systems combining multiple ML/DL algorithms to leverage their individual strengths. For instance, CNN-LSTM architectures have been applied to detect multi-stage attacks with promising accuracy and low false-positive rates.

4. Despite these advances, key challenges such as model explainability, real-time processing capability, and training time

still limit widespread deployment. This study builds upon the existing work by providing a side-by-side comparison under consistent experimental conditions.

Existing Method

1. Conventional IDS typically fall into two categories: signature-based and anomaly-based detection. Signature-based systems rely on predefined rules or patterns and are fast but limited to known threats.

2. Anomaly-based systems, on the other hand, detect deviations from normal behavior. They can detect zero-day attacks but tend to suffer from high false-positive rates, making them less practical in real-time settings.

3. Classical ML methods like Decision Trees, Logistic Regression, and Naïve Bayes have been widely used for intrusion detection, but they often require extensive feature engineering and struggle with non-linear or complex datasets.

4. DL models such as Autoencoders and CNNs have addressed some of these issues but demand higher computational resources and longer training times. Understanding the trade-offs between these methods is essential for practical deployment.

Proposed Method

1. This project proposes a structured performance analysis of ML (SVM, RF, DT) and DL (CNN, LSTM, CNN-LSTM) models using benchmark datasets. The goal is to determine which techniques offer the best balance between detection accuracy and computational efficiency.

2. Data preprocessing includes normalization, label encoding, and feature selection. SMOTE is used to balance the dataset where attack classes are underrepresented. Models are trained on the same data splits to ensure comparability.

3. Evaluation metrics include accuracy, precision, recall, F1-score, and AUC. Training time and resource consumption are also recorded to evaluate model scalability. Cross-validation is used to enhance result reliability.

4. Results are visualized using confusion matrices and ROC curves. Based on these insights, recommendations are made on which algorithms to use under specific deployment constraints such as low latency or high attack complexity environments.

RESULT

Now after all possible enhancements machine and deep learning algorithms are heading towards Federated learning as this federated learning provide data security, scalability and on time response availability. As we know machine or deep learning algorithms are dependent on dataset for training a model and this dataset has to upload to centralized server from local machines through internet and this data uploading may take huge network delay or latency for upload and this data will get exposed to centralized server and data security will be breached and due to network latency we may see delay in response also.

In some natural disaster scenarios like earthquake, floods, storm we need to have quick predicted response so government or peoples can take necessary action on time. So in propose work author applying Federated Learning for flood forecasting which allow local machines to train a model on local data and then upload only trained model to centralized server for global training and this technique avoid dataset upload which remove all existing barriers such as Latency, data breached and security. Local machine or centralized servers just have to take test data for prediction so network response and prediction will be quick.

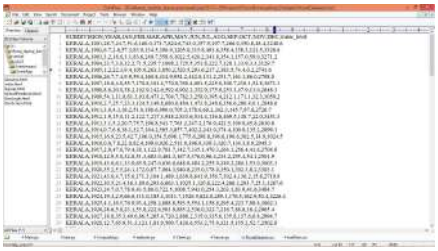
In propose work author using FFNN (feed forward neural network) algorithm to predict flood and this algorithm will trained dataset and then update

weights based on learning rate and training epochs. Epoch with less loss and high accuracy will have high weight and can predict label with high accuracy.

Propose work consists of following modules

1) The first step is onsite training and transmission of local data models using regional datasets towards central server for model aggregation.
2) The next step, global model is trained based on local modes that calculates multiple parameters and predicts the client station where flood is about to happen with 5 days lead time.
3) n the last step, local feed forward neural network (FFNN) model is trained on that specific client station to calculate expected water level and inform authorities for taking necessary actions regarding flood preparedness, mitigation and recovery.

In propose work author using 18 stations or rivers dataset to train FFNN algorithm locally and then report trained model to centralized server for global updates. Author has not published dataset on internet so we are using KERALA flood dataset from KAGGLE website. In below screen we are showing dataset details



In above dataset screen first row represents dataset column names and remaining rows represents dataset values where dataset has recordings of monthly rainfall and last column contains Water Level and based on predicted water level authorities will inform citizens about flood.

Extension Concept

In propose work author has used traditional Feed Forward neural network algorithms and did not used any advanced algorithms like Convolution2D Neural Network which gain popularity in all domains for its accurate and successful prediction accuracy of more than 90%. So to enhance accuracy we have used CNN2D as extension for flood forecasting.

Each algorithm performance is measure in terms of accuracy, MSE (mean square error) and RMSE (root mean square error) where MSE and RMSE refers to difference between True dataset value and predicted value, so the lower the difference the better is the algorithm.

We have designed this application as Window based project as this project has to upload trained model to centralized and JUPYTER will not give flexibility of model upload to server so we designed as window based application.

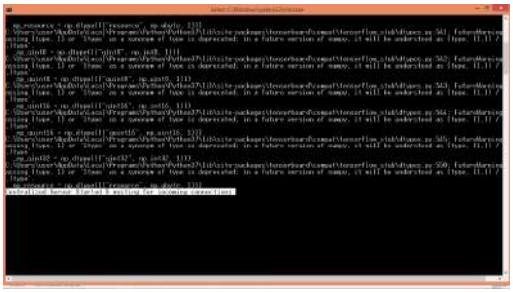
To implement this project we have designed following modules

1) Upload Flood Dataset: using this module we will upload, read and display dataset to application
2) Pre-process Dataset: using this module we will remove missing values, normalized and shuffle the dataset values
3) Train & Test Split: used to split dataset into train and test where application using 80% dataset for training and 20% for testing
4) Run Feed Forward Neural Network: this module used to trained FFNN algorithm by using train data as input and this trained model can be applied on test data to calculate prediction accuracy.
5) Run Extension CNN2D Algorithm: this module used to trained CNN2D algorithm by using train data as input and this trained model can be applied on test data to calculate prediction accuracy.
6) Upload Federated Model to Server: using this module locally trained models can be upload to centralized servers for global updates
7) Accuracy Comparison Graph: can be used to plot comparison graph between propose FFNN and extension CNN2D
8) Flood Forecasting using Test Data: can be used to upload test data and then extension

model will predict water level which help in knowing flood conditions.

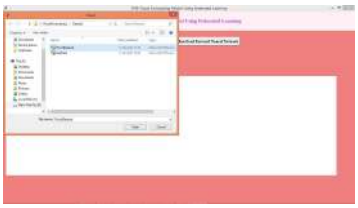
SCREEN SHOTS

First double click on 'runServer.bat' file to start centralized server and get below output



In above screen Centralized server started and now let it run and then double click on 'run.bat' file to start client which will train model locally by uploading local dataset and get below output

In above screen click on 'Upload Flood Dataset' button to load dataset and get below screen



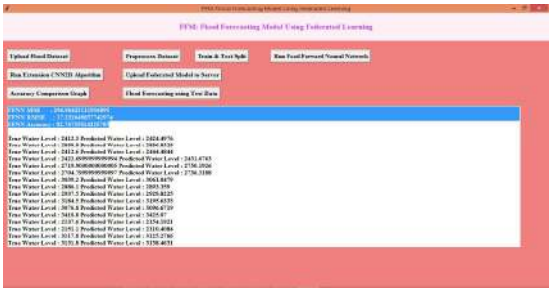
In above screen selecting and uploading 'Flood Dataset' and then click on 'Open' button to load dataset

In above screen dataset loaded and now click on 'Pre-process Dataset' button to process dataset and get below output

In above screen dataset pre-processing such as normalization and shuffling completed and now click on 'Train & Test Split' button to split dataset and get below output

In above screen displaying dataset size and then displaying train and test size and now click on 'Run Feed Forward Neural Network' button to train propose FFNN algorithm and get below output

In above screen FFNN training completed and in above graph x-axis represents Number of Days and y-axis represents Water level where red line represents True water level and green line represents Predicted water level and we can see both lines are fully overlapping with little gap so we can say predicted and true values are very close and FFNN giving best prediction and now close above graph to get below output

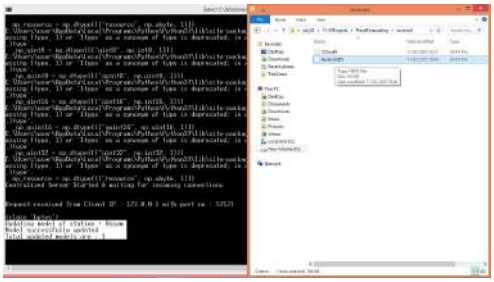


In above screen in first 3 lines we can see FFNN algorithm MSE, RMSE and accuracy values and then we can see true and predicted water levels for future days and now click on 'Run Extension CNN2D Algorithm' button to train extension algorithm

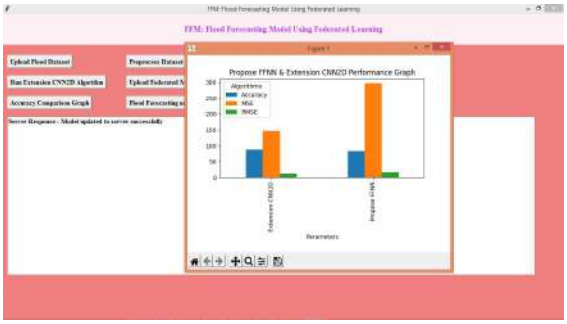
In above screen with extension we can see both predicted and true which means reads and green lines are fully overlapping so we can say extension model is better than propose and we can see MSE and RMSE also lower compare to propose and accuracy is high for extension algorithm and now close above graph and then click on 'Upload Federated Model to Server' button to upload trained model to server and get below output

In above screen just enter some station name and then click OK button to upload model to server and get below output

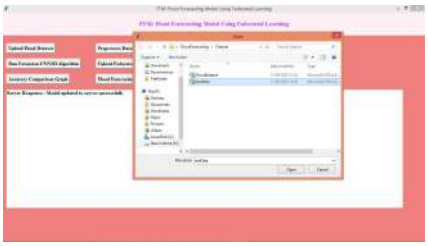
In above screen we got response from server as 'model uploaded' and in below server screen we can see received model details



In above screen in white colour text we can see server output about model saving and in server 'received' folder we can see 'Assam' model is saved and similarly for all given station server will saved model.



In above comparison graph x-axis represents algorithm names and y-axis represents accuracy and MSE values and we can see for extension algorithm accuracy is high and MSE, RMSE error is lower compare to propose FFNN algorithm and now close above graph and then click on 'Flood Forecasting using Test Data' button to upload test and then predict water level



In above screen uploading test data and then click on 'Open' button to get below output

In above screen before arrow symbol we can see test data and after arrow symbol =➔ we can see predicted water level

Conclusion

1. The study demonstrates that ML and DL algorithms provide a significant advantage over traditional rule-based systems for detecting web-based attacks. While DL models tend to outperform ML models in accuracy, they require more computational resources.

2. Among ML models, Random Forest showed strong performance across most datasets with relatively fast training times. Among DL models, CNN-LSTM hybrids provided the best results for complex and time-dependent attack patterns.

3. No single model excels in all scenarios; thus, the use of ensemble or hybrid systems can provide a more balanced solution. Real-world deployments should consider both performance and infrastructure limitations when selecting detection models.

4. Future work will explore online learning, adversarial attack resistance, and integration with real-time monitoring systems. The goal is to develop a robust, adaptive, and scalable IDS framework using the insights gained from this comparative study.