This is the accepted version of a paper presented at *IEEE iThings-2020*.

Citation for the original published paper:

Samikwa, E., Voigt, T., Eriksson, J. (2020)
Flood Prediction Using IoT and Artificial Neural Networks with Edge Computing
In:

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-427280
Flood Prediction Using IoT and Artificial Neural Networks with Edge Computing

Eric Samikwa  
Research Institutes of Sweden & University of Bern  
Switzerland  
eric.samikwa@inf.unibe.ch

Thiemo Voigt  
Uppsala University & Research Institutes of Sweden  
Sweden  
thiemo.voigt@ri.se

Joakim Eriksson  
Research Institutes of Sweden  
Sweden  
joakim.eriksson@ri.se

Abstract—Flood disasters affect millions of people across the world by causing severe loss of life and colossal damage to property. Internet of Things (IoT) has been applied in areas such as flood prediction, flood monitoring, and flood detection. Although IoT technologies cannot stop the occurrence of flood disasters, they are an exceptionally valuable apparatus for conveyance of catastrophe readiness data. Advances have been made in flood prediction using artificial neural networks (ANN). Despite the various advancements in flood prediction systems, there has been less focus on the utilisation of edge computing for improved efficiency and reliability of such systems. In this paper, we present a system for short-term flood prediction that uses IoT and ANN, where the prediction computation is carried out on a low power edge device. The system monitors real-time rainfall and water level sensor data and uses the temporal correlative information for ahead-of-time prediction of flood water levels using long short-term memory. The system can be deployed on battery-powered IoT devices. The results of evaluating a prototype of the system indicate that our model is suitable for real-time flood prediction. Furthermore, we obtain a low response time for running the ANN prediction analysis on the low-power edge device. The application of ANN with edge computing will help improve the efficiency of real-time flood early warning systems by bringing the prediction computation close to where data is collected.

Index Terms—Internet of things, flood prediction, artificial neural networks, edge computing, long short-term memory

I. INTRODUCTION

Floods are among the most common damaging natural disasters that affect millions of people across the world leading to severe loss of life and colossal damage to property, infrastructure and agriculture. According to the World Meteorological Organization, flooding remains the third biggest disaster in the world [1]. Due to climate change, scientists estimate a 4-inch sea level rise by 2030, which could potentially cause severe flooding in many parts of the world [2]. Based on a research conducted by the Institute of Environmental Studies, more than 60% of the world’s cities will be vulnerable to flooding in the next 30 years due to effects of the sea level rise [3].

Internet of Things (IoT) is a core technology being used in flood early warning systems. IoT characteristics provide effective guarantee for ahead-of-time perception and precaution, advance to reduce the impact of disasters [4]. Despite the fact that IoT technologies cannot stop the occurrence of disasters, they are an exceptionally valuable apparatus for conveyance of catastrophe readiness data. Such data can be used for geographical flood simulation modeling [5], which aids in policy making in flood disaster risk management.

For real-time flood early warning systems, information delivery is key. Thus, there is a need to ensure that information delivery must be concise, right to the point, usable and in a timely manner. There are several factors that are attributed to the efficiency and effectiveness of real-time early warning systems for floods. These include the correctness of prediction of a flood occurrence, the amount of time needed to make a prediction, the reliability of the communication networks used in the early warning system as well as the deployment and maintenance cost of the systems. One major challenge for the future of IoT applications is that all data collected from end devices, such as IoT sensors has to be sent via the Internet to a central processing platform. This creates a scalability problem in IoT as there is an exponential growth of IoT devices which connect to the Internet for either receiving information from the cloud or delivering data back to the cloud [6].

Timeliness is an important factor in short-term flood prediction. Information about the occurrence of a flood disaster needs to be presented to the affected areas as early as possible. Collecting data to a central processing platform may take a considerable amount of time especially for real-time flood prediction systems. Most of the deployments of IoT sensors that collect information that is used for prediction of flood events are located close to water bodies that may be far from the cities and maintaining a good Internet connectivity at all times may lead to costly deployments. The reliability of the early warning systems is another issue to consider. Most flooding events occur in unpleasant weather conditions such as heavy rainfall or storms which can affect Internet connectivity. We need mechanisms that enable us to process the sensor data within the proximity of where it is collected, both logically and physically.

Since most IoT devices are battery powered or operate on low power, the mechanism for processing the data at the edge will be much more effective when the processing platform also operates on low power. That means the implementation of the prediction algorithms, software, and communication mechanisms are suitable for resource-constrained devices. Despite the various advancements in flood prediction systems through the use of artificial intelligence methods, there has
been less focus on the utilisation of edge computing for improved efficiency and reliability of such systems. In this paper, we present a system for real-time flood prediction that utilises IoT sensing and artificial neural networks (ANN), where processing of the sensor data is carried out on a low-power edge device. The system monitors real-time rainfall and water level time series and uses the temporal correlative information for ahead-of-time prediction of flood water levels using long short-term memory (LSTM). LSTM is a type of ANN suitable for handling time series data [7]. We evaluate our model on a real dataset using statistical error measures for regression. We implement a prototype dissemination system for sending flood early warning messages via Twitter. We measure the response time for reading sensor data and running ANN analysis on the resource-constrained device and show that our approach is efficient enough to support the timeliness of the early warnings.

The main contribution of our work is an approach to ahead-of-time flood prediction using an ANN at the edge. While previous solutions have been designed to run the prediction computation in the cloud, our solution utilises a low-power edge device for real-time ANN prediction analysis. This improves the timeliness and reliability of our system as it avoids the need to send raw sensor data over the Internet in real time for on-the-fly flood prediction. The second contribution of our work is a better short-term flood prediction model accuracy compared to similar real-time flood prediction systems. This is achieved through the use of an LSTM.

Section II provides a theoretical background in edge computing and the ANN model used in this work. Section III presents other flood prediction systems that are related to our work. In Section IV we present our approach to real-time flood prediction. We evaluate our system in Section V. Lastly, in Section IV we present the conclusions of our work.

II. THEORETICAL BACKGROUND

A. Edge Computing

Edge computing is defined as a part of a distributed computing topology in which information processing is located close to the edge [8]. Edge computing brings computation closer to the devices where information is collected, rather than relying on a central location. IoT edge devices generate enormous amounts of data at a fast velocity during the course of their operations. The data generated from the distributed IoT devices is transmitted to a remote cloud platform for processing via the Internet in conventional IoT architectures. The process of transferring the big data is expensive, consuming a huge amount of bandwidth, energy, and time [9]. Hence the need to design efficient data processing architectures to explore the valuable information in real time at the edge. Edge devices especially those that are used in IoT applications are characterised by low power [10].

There are a number of parameters to consider when determining an edge computing platform for an application [11]. These include power consumption of the edge device, computation time of the application on the edge device, proximity of the edge device to the end devices, and the access medium between the edge device and the end devices.

B. Recurrent Neural Networks (RNN)

RNN is a class of ANN where connections between nodes form a directed graph along a temporal sequence [12]. RNNs uses their internal state to process variable length sequences of inputs there by exhibiting memory in the model. When unrolled in time, an RNN can be considered as a deep neural network with indefinitely many layers.

Figure 1 shows an RNN that has been unfolded in time, where \(x_t\) and \(\hat{y}_t\) are the input and output at time \(t\) respectively, and \(h_t\) is the value of the hidden state at time \(t\).

This dynamic system is defined by the equations:

\[
\begin{align*}
\dot{h}_t &= \Phi_s(w^T h_{t-1} + u^T x_t) \quad (1) \\
\hat{y}_t &= \Phi_o(v^T h_t) \quad (2)
\end{align*}
\]

where \(w\), \(u\) and \(v\) are respectively the transition, input and output matrices, and \(\Phi_s\) and \(\Phi_o\) are element-wise nonlinear activation functions such as ReLU for \(\Phi_s\) and sigmoid for \(\Phi_o\).

LSTM is a special type of recurrent hidden unit, capable of learning long-term dependencies [7]. RNN may suffer from the vanishing/exploding gradients problem as the gap between the relevant observed information and the desired prediction grows. LSTM was introduced to solve this problem and has since been very useful in practical time series predictions. Figure 2 shows the internal structure of an LSTM cell.

An LSTM’s state is split in two vectors, namely short-term state \(h\) and long-term state \(c\). LSTM can remove or add information to the cell state, regulated by the three gates namely forget gate \(f_t\), input gate \(i_t\), and output gate \(o_t\).
III. RELATED WORK

Satria et al. [15] have developed a prototype of an IoT-based flood monitoring system that provides users with real-time water levels and rain weather conditions. Water level and rainfall sensor data is transferred over WiFi to an Arduino Uno where a comparison with thresholds is run to determine threat levels and the resulting flood status is displayed using a web browser. Their prototype does not include more processing of the collected data to determine the threat level ahead of time as it mainly targets at displaying current water level and rainfall status. Furthermore, their system requires Internet connectivity at all times and the hardware platforms they use are power-hungry.

Bande et al. designed a flood prediction system using IoT and ANN [16]. Their system monitors humidity, temperature, air pressure, rainfall and water level for prediction of a flood occurrence ahead of time. The sensors are attached to a Raspberry Pi which communicates over WiFi. The data is sent in real time over the Internet to a cloud-based ANN for flood prediction and the results are displayed in a web-based platform. They use a nonlinear autoregressive network with exogenous inputs (NARX) for the prediction of flood water levels. Their results show that their system provides a one-time-step ahead-of-time prediction with an accuracy of 0.88.

Abdullahi et al. [17] developed an IoT-based intelligent flood disaster warning system, where they utilised a cloud-based Microsoft Azure ML pre-trained 2-class neural network for on-the-fly prediction of flood status. They evaluate their system in a controlled environment where they use a miniaturized flow, water level, and pressure sensors connected to a NodeMCU with WiFi access. Their system provides on-the-fly flood prediction using the real-time sensor data with the highest accuracy of 0.98 for a controlled environment. Their system was evaluated using rules and generated data that was suited for the controlled environment and the prediction did not use a time-series prediction model.

Jayashree et al. proposed a flood early warning system that is not dependent on mobile towers for alert message broadcasting [18]. Their system does not require Internet as it is based on ZigBee. A dam water filling plant consists of a distance sensor and a flow sensor that send real-time data to a local server. If the level exceeds a specified limit, alert messages are broadcasted through ZigBee. The system was designed for monitoring of a flood occurrence on a dam and provides real-time alerts in case of a flooding event while operating on low power. However, the system does not include more data analysis for ahead-of-time prediction of a flooding event.

IV. EDGE-BASED FLOOD PREDICTION

We present our system for on-the-fly ahead-of-time prediction of flood water levels. The system uses IoT sensors to collect environmental parameters in real time and makes a short-term forecast of the water level using an ANN at the edge. Similar to the system by Bande et al. [16], environmental parameters such as water level and rainfall are utilised. Additional environmental conditions that affect rainfall such as humidity, temperature, air pressure may improve prediction of floods when using ANN. However, rainfall and water level data have proven to be adequate in the prediction of flood water levels using time series ANN analysis with good performance [19] [14]. Similar to the system by Abdullahi et al. [17], our system is designed for monitoring sensor data in real-time for short-term flood prediction on the fly.

Unlike both systems, our system utilises edge computing as the flood prediction is carried on a low power device within the IoT wireless sensor network (WSN). This means the raw sensor data is not sent over the Internet for the real-time flood prediction but rather the output of the prediction at relevant intervals. Secondly, our system operates on low power as it uses low-power devices for IoT sensing and edge computing, and utilises ultra-low power WSN communication, BLE [20], between the sensing nodes and the edge computing device. Thus, the system can be deployed as a battery-powered IoT flood prediction system where Internet access is provided to the edge computing platform and not to individual sensor nodes. Lastly, our system uses LSTM for prediction using rainfall and water level time series. Our system is, however, ready to be extended to more sensing modalities. We evaluate our model on a real dataset [21].

A. System Design

We use distance and rainfall sensors to collect water level and rainfall data respectively. The sensors are connected directly to a low power IoT BLE device, Arduino Nano 33 BLE. The combination of the two sensors and the Nano 33 BLE makes a single IoT sensing node. A sensing node communicates to an edge device, Raspberry Pi 3B+, via BLE.

B. IoT Framework

Multiple Arduino Nano 33 BLEs can be connected to a Raspberry Pi through BLE connections as BLE supports multiple and mesh connections. The sensor data is processed on the edge device, Raspberry Pi. Depending on the output of the forecast, early warning messages can be disseminated through the Internet, local networks or an alarm system directly connected to the edge computing device. BLE is suitable for our system since both the IoT nodes and the edge device are low-power devices and can be deployed on battery power. BLE is more energy-efficient in terms of the number of bytes transferred per Joule spent than WiFi or Zigbee [22]. Although BLE offers lower data rates than WiFi, it is adequate for the amount of data being sent from the sensors, i.e., only rainfall and water level data.

For this study, we implement a prototype dissemination system using ThingSpeak [23] to send out early warning messages via Twitter. Other early warning message dissemination techniques that can be used for our system include short message service (SMS) and local IoT WSN [18]. Figure 3 shows the block diagram of two IoT sensing nodes and the edge computing platform in our system.
Fig. 3. Block diagram of two IoT sensing nodes and edge computing platform for our system. Early warning messages are sent via Twitter in case of predicted threat water levels.

C. Short-Term Flood Prediction Using LSTM

Short-term flood prediction is defined as the prediction of a flooding event in real time between a few minutes and hours preceding the flood [24]. In contrast to long-term flood prediction that is mostly used for policy analysis purposes, the aim of short-term flood prediction is to reduce immediate damage and harm caused by a flood disaster. Short-term flood prediction systems with lead time are considered important research challenges, particularly in highly populated areas for timely warnings to residences [25]. Despite several improvements in statistical weather prediction models, artificial intelligence and machine learning methods, short-term flood prediction remains a challenging task [26][27].

In this paper, we utilise an LSTM for short-term prediction of flood water levels. By observing a multivariate time series of rainfall and water level, the model outputs a forecast of the water level ahead of time. The input to the model is ten time step (i.e. ten hours) sequences of water level and rainfall data and the output is a forecast of the water level in the next ten hours. Figure 4 shows the architecture of our model, where $\hat{X}$ and $\hat{y}$ represents the multivariate input and the output respectively.

We use a ReLU activation function in the LSTM layer for activation. A ReLU is a unit in neural networks that uses the activation function $max(0,x)$ [28]. Thus, the neurons will only be activated if the output of the linear transformation is not less than 0. The main advantage of using the ReLU function over other activation functions is its computational efficiency. ReLUs are computationally more efficient to compute than other activation functions since ReLUs just need to pick $max(0,x)$ and not perform expensive exponential operations as in Sigmoid. Besides having better accuracy than the sigmoid and tanh activation functions, a ReLU activation function is a suitable choice for this system as our model runs on a resource-constrained edge device where the model’s computational efficiency is relevant.

We add the dropout layer in order to prevent overfitting during training. Dropout is a technique for improving the generalization error of neural networks [29]. Dropout prevents overfitting and produces far more robust models by adding noise during training without slowing down the learning process. Dropout is not used during testing to allow all the trained units to contribute to the prediction process. The last layer in the neural network is the output dense layer.

The model is compiled with a MSE loss function. The loss function is used to optimize the parameter values in the model during training. This is done by mapping a set of parameter values for the network onto a scalar value that indicates how well those parameters perform. MSE is calculated as the average of the squared differences between the predicted and actual values as shown in Equation (3).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2$$

The model is compiled with the Adam optimiser. The Adam optimisation algorithm combines the heuristics of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [30]. As AdaGrad, it keeps track of an exponentially decaying average of past gradients and as RMSProp it keeps track of an exponentially decaying average of past squared gradients. Adam computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients. Adam works well in practice and compares favorably to other stochastic optimization methods [30].

We use TensorFlow Lite for real-time ANN analysis of the sensor data on the edge device. TensorFlow Lite enables on-device machine learning inference with low latency and a small binary size and is hence suitable for embedded and IoT devices.

V. EVALUATION

In this section we evaluate our system with respect to forecast accuracy and response time and introduce the dataset we use in this paper.

A. Dataset Description

We use a dataset consisting of hourly rainfall and water level data sourced from Melbourne Water [21]. The data is in the form of a time series, represented by time-stamps with corresponding rainfall and water level values. The dataset consists of 78844 rows and was collected hourly between
2009-12-01 00:00:00 and 2018-12-02 18:00:00, representing approximately 9 years of hourly rainfall and water level data.

### B. Forecast Accuracy

We study the accuracy of the forecasts produced by the LSTM model and how it is impacted by the number of time steps between the observed time series and the predicted water level. We expect a higher accuracy when predicting the water level for a few time steps ahead and reduction in accuracy as the number of time steps increases. Performance of a regression model, such as flood water level prediction model, can be evaluated using different statistical error measures. The statistical error measures include RMSE and R.

- **RMSE**: The RMSE measures the square root of the average of the squared difference between the predictions and the ground truth.

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2}
\]  

- **R**: R is a statistical measure of the strength of the relationship between the relative movements of two variables. R values range between -1.0 and 1.0, where -1.0 shows a perfect negative correlation, 1.0 shows a perfect positive correlation. R value of 0.0 shows no linear relationship between the movement of the two variables.

\[
R = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2(\hat{y}_i - \bar{\hat{y}})^2}}
\]

We evaluate our model based on these metrics for predictions of water levels at different time steps ahead on unseen test data. The test data is extracted from the most recent part of the dataset before data pre-processing and is not used during the training and validation of the model.

For performance comparison we implement a regression model using feed forward neural network (FFNN) and test it using the same dataset. Table I presents RMSE and R for FFNN and the LSTM model at different time steps ahead predictions. The results show higher R values and low RMSE values for the LSTM model compared to the FFNN model. This indicates that the LSTM model is better than the FFNN model at forecasting the flood water levels. This can be attributed to the ability of LSTM to store sequence information due to its memory capability.

Figure 5 and Figure 6 illustrate the forecasts produced by the LSTM model for the water level one time step ahead and ten time steps ahead respectively plotted together with true water levels. The figures show that the best performance is seen when predicting one time step ahead where the predicted data points almost overlap with the true data point at all times. The results of the evaluation for each prediction at different time steps are plotted to visualise how well the model performs for variable number of time steps ahead in the predictions. Figure 7 shows the R values for the LSTM model performance at different time steps ahead prediction of water levels.

<table>
<thead>
<tr>
<th>time steps</th>
<th>FFNN RMSE</th>
<th>FFNN R</th>
<th>LSTM RMSE</th>
<th>LSTM R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.082</td>
<td>0.869</td>
<td>0.038</td>
<td>0.976</td>
</tr>
<tr>
<td>2</td>
<td>0.090</td>
<td>0.858</td>
<td>0.048</td>
<td>0.962</td>
</tr>
<tr>
<td>3</td>
<td>0.103</td>
<td>0.837</td>
<td>0.060</td>
<td>0.941</td>
</tr>
<tr>
<td>4</td>
<td>0.109</td>
<td>0.808</td>
<td>0.063</td>
<td>0.933</td>
</tr>
<tr>
<td>5</td>
<td>0.111</td>
<td>0.791</td>
<td>0.067</td>
<td>0.918</td>
</tr>
<tr>
<td>6</td>
<td>0.121</td>
<td>0.790</td>
<td>0.075</td>
<td>0.878</td>
</tr>
<tr>
<td>7</td>
<td>0.139</td>
<td>0.772</td>
<td>0.084</td>
<td>0.869</td>
</tr>
<tr>
<td>8</td>
<td>0.145</td>
<td>0.760</td>
<td>0.094</td>
<td>0.843</td>
</tr>
<tr>
<td>9</td>
<td>0.152</td>
<td>0.753</td>
<td>0.099</td>
<td>0.823</td>
</tr>
<tr>
<td>10</td>
<td>0.167</td>
<td>0.749</td>
<td>0.106</td>
<td>0.806</td>
</tr>
</tbody>
</table>

Fig. 5. One hour ahead-of-time forecasts of water levels together with true water levels. The similarity is very high (R=0.976).

The highest R value is 0.976 obtained for predictions of water levels one time step ahead. This means there is 97.6% similarity between the predicted and the true water levels. This can also be seen from the visualisation in Figure 5 and the lowest RMSE value of 0.038 obtained for one time step ahead prediction. This indicates the suitability of the LSTM model for short-term flood prediction by observing rainfall and water level time series. This is a better performance compared to a similar flood prediction system by Bande et al. [16] that uses NARX RNN for flood prediction on a dataset with similar features, with an R value of 88.12% for one time step ahead prediction. We could not get access to the exact dataset they used in their project.

The R values decrease gradually each time when the number of time steps ahead in the predictions increase, with a value of 91.8% and a value of 80.6% at five and ten time steps ahead predictions respectively as shown in Figure 7. The RMSE values increase gradually each time when the number of time steps ahead in the predictions increase as shown in Table I. This is as expected since the chronological connection between the observed time series and the predictions is weaker when the gap between them increases.
C. Response Time

The response time consists of two parts. Firstly, the time taken for the Raspberry Pi to establish a BLE connection to the Nano 33 BLE and read sensor data over the connection. Secondly, the time taken to prepare the raw sensor data for ANN prediction, making predictions using the LSTM model, and making comparisons to thresholds to determine the flood alert status.

In order to ascertain the value of the response time, we run the system multiple times and calculate the average of the response times recorded for each run. The average time for establishing a BLE connection and reading sensor data over the connection is 5 seconds and the average time for data preparation and forecasting is 1.3 seconds. Thus, the flood prediction system completes the process in less than 7 seconds. This is a sufficiently low response time considering that the shortest time span for a prediction in our model is one hour ahead of time. Computation time is a key factor to consider when choosing an edge computing platform [11]. This shows that the edge computing platform in our system is able to handle the computational requirements of the application without compromising the integrity of the outcome.

D. Comparison of Alternative Design Approaches

We present a comparison of the design alternatives we considered for our system. We examine the size of the model when compiled by Tensorflow and Tensorflow Lite to gain insight on the model size reduction. We compare the performance of the LSTM model when using different activation functions and optimizers.

1) Model Size Reduction by Tensorflow Lite: Figure 8 shows the model sizes obtained by compiling the model using Tensorflow and Tensorflow Lite, before and after training. The figure shows there is almost a 60% decrease in model size when using Tensorflow Lite. This is a necessary optimisation as our system runs on a resource-constrained low-power device.

2) Performance Comparison of ReLU, Sigmoid and Tanh Activation Functions: We examine our model for accuracy with ReLU, sigmoid and tanh activation functions. Figure 9 shows the R values for the comparison of ReLU, sigmoid and tanh activation functions. The model performs better when using ReLU activation function as compared to using sigmoid or tanh consistently for predictions at different time steps ahead. This is expected since ReLU converges faster and, in contrast to sigmoid or tanh, does not suffer from the vanishing gradient problem [28].

3) Performance Comparison of Adam and SGD Optimizers: We examine our model for accuracy with Adam and SGD optimizers. Figure 8 shows the R values for the comparison of Adam and SGD optimizers. The figure shows that the model performs better when using the Adam optimizer as compared to using SGD consistently for predictions at different time...
steps ahead. This is as expected since Adam combines heuristics of both RMSprop and SGD with momentum for adaptive learning rate optimization [30].

Fig. 10. Performance comparison of Adam and SGD optimizers. Adam outperforms SGD.

VI. CONCLUSIONS

In this paper we present a system for on-the-fly short-term flood prediction. The system monitors real-time rainfall and water level time series and uses the temporal correlational information for ahead-of-time prediction of flood water levels using LSTM. The prediction accuracy of our model was evaluated for predictions of flood water level at different time steps ahead. The results indicate that our model is suitable for real-time flood prediction. Furthermore, we obtain a low response time for running the ANN prediction computation on the low power edge device. The most distinct feature of our system is that the predictions are carried out on a low-power edge computing device. This means the raw sensor data is not sent over the Internet in real time for on-the-fly flood prediction but rather only the result of the prediction when necessary. To achieve this efficiently and without delay, we utilised a low computationally demanding deep learning platform, Tensorflow Lite, and low power IoT wireless communication technology, BLE. Thus, our system can be deployed as a battery-powered IoT flood prediction system with low deployment and maintenance costs. The application of ANN with edge computing will help improve the efficiency and reliability of flood early warning systems by bringing the prediction computation close to where data is collected.

REFERENCES