Earth and Space Science

RESEARCH ARTICLE
10.1029/2022EA002385

An End-To-End Flood Stage Prediction System Using Deep Neural Networks

L. Windheuser[1], R. Karanjit[2], R. Pally[2], S. Samadi[3], and N. C. Hubig[2]

[1]Department of Informatics, Technical University of Munich, Munich, Germany, [2]School of Computing, Clemson University, Clemson, SC, USA, [3]Department of Agricultural Sciences, Clemson University, Clemson, SC, USA

Key Points:

- A fully automated end-to-end flood stage image detection system is developed using two US Geological Survey gauging data
- Compared to other models, Long Short-Term Memory predicted the flood stage more accurately during both historical and real-time events
- A longer lead time flood stage forecast requires more physics-based features to be incorporated into the Deep Neural Network models

Correspondence to:
S. Samadi,
samadi@clemson.edu

Citation:
Windheuser, L., Karanjit, R., Pally, R., Samadi, S., & Hubig, N. C. (2023). An end-to-end flood stage prediction system using deep neural networks. *Earth and Space Science*, *10*, e2022EA002385. https://doi.org/10.1029/2022EA002385

Received 18 APR 2022
Accepted 6 DEC 2022

Author Contributions:

Conceptualization: S. Samadi
Data curation: L. Windheuser, R. Pally, S. Samadi
Funding acquisition: S. Samadi
Methodology: S. Samadi, N. C. Hubig
Software: L. Windheuser, R. Karanjit, S. Samadi
Supervision: N. C. Hubig
Validation: L. Windheuser, R. Karanjit, R. Pally
Writing – original draft: L. Windheuser, S. Samadi
Writing – review & editing: R. Karanjit, S. Samadi

Abstract The use of automated methods for detecting and classifying different types of labels in flood images have important applications in hydrologic prediction. In this research, we propose a fully automated end-to-end image detection system to predict flood stage data using deep neural networks across two US Geological Survey (USGS) gauging stations, that is, the Columbus and the Sweetwater Creek, Georgia, USA. The images were driven from the USGS live river web cameras, which were strategically located nearby the monitoring stations and refreshed roughly every 30 s. To estimate the flood stage, a U-Net Convolutional Neural Network (U-Net CNN) was first stacked on top of a segmentation model for noise and feature reduction that diminished the number of images needed for training. A Long Short-Term Memory (LSTM), a dense model, and a CNN were then trained to predict the flood stage time series data in near real-time (6, 12, 24, and 48 hr). The results revealed that the U-Net CNN has a higher accuracy for image segmentation if the algorithm is stacked in front of the network. The absolute error with the U-Net was 0.0654 feet at the Columbus while it was 0.0035 feet at the Sweetwater Creek, which were practically low for flood stage estimation. For time series prediction, among three models, the LSTM predicted the flood stage values more accurately during both historical (2015–2022) as well as real-time forecasts, particularly for 24 and 48 hr timescales. We extensively evaluated the proposed flood stage prediction system against current state-of-the-art methodologies partly crowd-sourced and mined in real-time.

Plain Language Summary In the past few years, image processing techniques are used for image labeling tasks given their capacity to learn rich features. Real-time river stage prediction is the subject of numerous studies of a similar nature. Still, they have yet to combine multiple datasets (such as time series and image data) for flood stage prediction. Here, we examined how Convolutional Neural Network, Long Short-Term Memory, and a dense model can be applied to stream live images from the US Geological Survey (USGS) web cameras and label the features for real-time flood gauge estimation. The preliminary motivation for using these models was to explore the strength of different types of representations for predicting class labels and estimating flood stages in real-time. We evaluated our models on a new image data set that was collected from multiple rivers scraped from the USGS live webcams with their associated annotated labels. Compared to other techniques, the proposed end-to-end flood stage estimation approaches produced state-of-the-art results for the two USGS stations, and also demonstrated the capability of using data intelligence tied to different sources of labeling in improving flood stage estimation.

1. Introduction

Floods are on the rise globally with the frequent record-breaking events occurring during the past few years in the US alone. These extreme events pose a considerable threat to human life and result in destructive damage to property, communities, and the built environment (e.g., Phillips et al., 2018). The south and the southeast US have experienced frequent storms with annually, on average, more than 85 named and unnamed thunderstorms (NWS, 2020). These events happened in quick succession (~2 weeks apart) and produced catastrophic flooding in wide geographic areas (~1,000 km swath) and within short timespans (less than a 48-hr period; Donratanapat et al., 2020). Successive flood events can even lead to higher costs in terms of repairing and rebuilding destroyed buildings and critical infrastructures (CIs) due to a lack of early warning systems (e.g., Donratanapat et al., 2020; Field et al., 2012; Hinkel et al., 2014). This necessitates the importance of detecting flood magnitudes ahead of the event to protect communities and CIs. The flood stage is the height of the water surface in a stream gaging station, not the height throughout the stream. A vast amount of research has been conducted to develop different tools and test their reliability in predicting near real-time flood stage estimation (Krzysztofowicz et al., 1994).

However, these tools require high computational time and effort as well as different sources of spatial and physics-based data to be trained and validated which is time-consuming and not practical for real-time application (e.g., Bermúdez et al., 2018; Hubig et al., 2017; Ivanov et al., 2021; Zahura et al., 2020).

Time-lapse images and videos driven from surveillance cameras can provide a vast amount of big data that can be used to constrain intelligent techniques and provide valuable insights into flood risk and severity. These real-time surveillance cameras have been strategically installed by several federal agencies, such as the US Geological Survey (USGS) across numerous river networks to meet the need for timely assessment of riverine flood situations (Dazzi et al., 2021). The real-time videos and images can be used to track the increasing flood stage during a storm and continuously monitor the potential impacts of flooding on nearby communities and CIs (see Bhola et al., 2019; Moy et al., 2019). Videos and time-lapse images can also be processed to extract image frames and related information and measure a range of flood characteristics such as flood stage and inundation areas. Using the webcam images as well as historical weather data, one can develop an intelligent flood image label detection system to monitor and evaluate riverine flood conditions in near real-time.

Deep learning algorithms are exceptionally valuable tools for collecting and analyzing catastrophe readiness and countless flood image data. Convolutional Neural Networks (CNNs) are one form of deep learning algorithm widely used in computer vision that can be utilized to study flood images and assign learnable labels to various objects in the images (e.g., Pally & Samadi, 2022). Additionally, Recurrent Neural Networks architectures such as Long Short-Term Memory (LSTM) have demonstrated great success in sequential label simulation and time series prediction tasks (Kratzert et al., 2018; Tabas & Samadi, 2022; Shen, 2018). Despite the widespread use of these intelligent techniques, very few studies have focused on using them in flood assessment. Paul and Das (2014) were among the first scholars who used CNN for the Manu River level prediction in India. They forecasted the river level in advance (up to 72 hr) using precipitation and observed flood stage as input features. They also used a Feed Forward Neural Network and reached a mean absolute percentage error (MAPE) of 6.5% for 24 hr, 23.7% for 48 hr, and 47.4% for 72 hr forecasts. Moy et al. (2019) used a deep CNN (DCNN) to detect floodwater in surveillance footage and a static observer flooding index (SOFI) as a proxy to estimate flood stage fluctuations visible from a surveillance camera's viewpoint. Their results revealed that DCNN and SOFI have the potential to be applied to a variety of surveillance camera models and flooding situations without the need for on-site camera calibration. Bhola et al. (2019) employed deep learning and edge detection techniques to identify the water surface in an image. They showed that deep learning and edge detection techniques can be used as additional point source validation information to substantially improve flood inundation forecasting. Jiang et al. (2019) used a CNN approach for waterlogging depth estimation from video images based on reference objects. They achieved a MAPE of 19.97% in the testing period. Other water depth estimation studies used LiDAR data with high accuracy. Two excellent applications of CNN-LiDAR image label detection approaches are given by Hilldale and Raff (2008) and Tonina et al. (2019).

Recently, Baek et al. (2020) used a CNN to predict flood stage across the Nakdong River Basin in South Korea. They achieved a high precision score with the mean square error (MSE) of 0.001 m for stage prediction. Faruq et al. (2020) employed the LSTM and Radial Basis Function Neural Network to simulate the flood stage for the Klang River in Malaysia. Their training data consisted of historical flood stage only, disregarding any further weather features. In their study, the LSTM-RBFNN model performed a root mean square error (RMSE) value of 0.02 and a coefficient of determination (R^2) value of 0.98. Their results verified that the LSTM network with a specified training set provided a promising alternative technique to the solution of flood modeling and forecasting problems. In a sequence, Dazzi et al. (2021) evaluated the accuracy of machine learning models to predict the gauge height data of the Parma River at the Colorno gauging station in Italy. They applied upstream stage observation along with downstream gauging data for possible backwater to feed into machine learning models. They tested a Support Vector Regression (SVR), a Multi-Layer Perceptron, and an LSTM network. More recently, Al-Fawa'Reh et al. (2021) used Random Forest, Decision Tree (DT), Linear Regression, SVR, and the K-Nearest Neighbor models to predict the flood stage for the Jordan River in Jordan. They fed the relative humidity, daily rainfall, wind speed, and air temperature into the models to predict the flood stage data. A large data set was incorporated into these models that were collected from the past 38 years across 13 stations in the region. Their analysis revealed that the DT model is the most skillful approach, which was able to forecast up to 24 hr in advance accurately with a low mean absolute error (MAE) of 0.021 m.

Despite these studies, the development of neural network algorithms for real-time implementation that support multiple modeling hypotheses is, however, still in its nascence, and yet has not provided the systematic approach

that is needed to combine multiple datasets, that is, time series and image data. In this research, we developed multiple neural networks for real-time flood stage prediction and segmentation using both image and time series data from two USGS gauging stations that is, the Columbus and the Sweetwater Creek, Georgia (GA), USA. Here, we presented a novel and practical method that exploits flood stage images and treats flood stage forecasting as regression problems with appropriate loss functions and metrics for result inspection. By developing and testing multiple neural network algorithms for real-time flood stage prediction, this research hopes to provide an end-to-end system designed for deployment in large, gauged rivers, implying flood stage data as well as river webcam images. Using proposed approaches, this study addresses three research questions notably; (a) How can a combined image and time series data provide the intelligence for historical and near real-time flood stage prediction? (b) What forecast lead time (6, 12, 24, and 48 hr) can be predicted in advance with high accuracy compared with others? And (c) what type of neural network algorithm can accurately simulate both short- and long-term fluctuations in the data? By answering these questions, we aim to provide a tool that can use time series data and exploit existing surveillance infrastructure to address flooding issues in at-risk areas and support flood modelers and decision-makers. The main contributions of this paper include the following:

- Developing a novel approach for estimating flood stage data by stacking a CNN on top of a segmentation model as a pre-processing step for noise and feature reduction. The stacked segmentation model was designed to consistently deliver the most accurate and stable predictions by combining different individual strengths of CNN and LSTM.
- Developing a three-layer stacked dense neural network connected by an encoder-decoder architecture as a fully connected feedforward neural network for flood stage forecast.
- Predicting flood stage based on historical weather data using different architectures for up to 12 hr in advance.
- Creating a custom flood image data set scraped in real-time from the USGS live river webcams and their respective gauge height data as labels. This data set was used for real-time gauge height estimation and segmentation.

The remainder of this paper is structured as follows: Section 2 includes the case studies of the chosen rivers as well as includes a description of the data set used in this research. Section 3 presents the methodology and algorithms as well as performance metrics to compare the results and the accuracy of prediction. Section 4 presents the results of multiple experiments for different datasets. Lastly, the conclusion of this study is presented in Section 5.

2. Case Studies and Data

2.1. Case Study 1: Chattahoochee River at Columbus, GA

The Chattahoochee River at Columbus, GA (USGS02341460) is 4,630 square miles (Hydrologic Unit Code 03130003). It formed in the lower half of the border of GA and Alabama (AL), USA. This gauge is equipped with both river and climate stations instrumented by the USGS. Different datasets such as water temperature, air temperature, air pressure, wind speed, wind direction, precipitation, relative humidity, discharge, and gauge height are also monitored at this gauging station. For most measurements, however, the collection started just recently, and gauge height data were long enough to train models. We used the precipitation data of this station as an input variable for neural network algorithms. The precipitation data has been monitored since 31 July 2013, with a 15-min time interval which was used as an input variable for constructing neural networks. In addition, the flood stage images were gathered from a USGS live river webcam that is located on the Chattahoochee River at Columbus (02341460) in proximity to the flood stage gauging station.

2.2. Case Study 2: Sweetwater Creek Near Austell, GA

The Sweetwater Creek is a stream with a length of 73.4 km in GA near Atlanta and is a tributary of the Chattahoochee River. The Sweetwater Creek has a stream gauge near Austell, GA (USGS 02337000), where the USGS has regularly monitored precipitation, discharge, and gauge height data since 2010 in 15 min increments. A river webcam is also located, at Sweetwater Creek near Austell water-monitoring site (02336910). Only the gauge height data was used as input data because precipitation data for this station was recorded from October 2012 to 2017 with significant missing values that were too short to incorporate into the models for flood stage simulation during 2015–2022. We comparably selected the same time period as the Chattahoochee River with >240,000 rows of data.

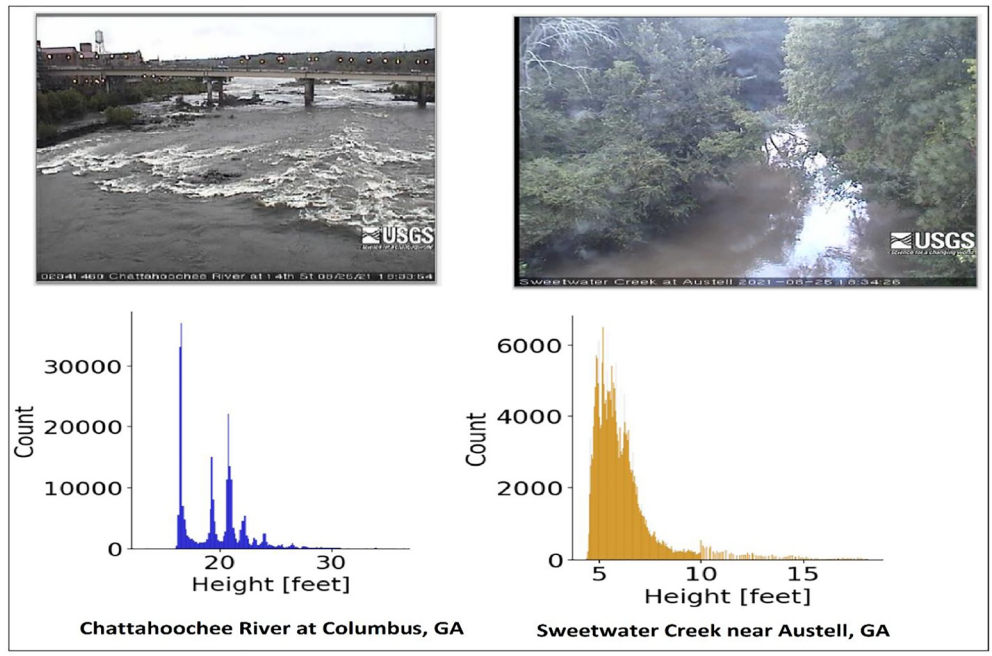


Figure 1. The location of US Geological Survey (USGS) live river webcams and their associated gauging stations (photos are taken on 26 August 2021). Distribution plots of the observed gauge heights (in 15 min increments) are displayed for each station from 1 January 2015 to 2 September 2022.

To fill the missing values in the data, we used a linear interpolation technique (spline interpolation method) using the Python pandas *DataFrame.interpolate()* method. We applied this method to time series data, although the missing values were insignificant (less than 1%). Using the USGS live river webcam images, we were able to scrape around 300 pictures and their respective gauge heights for training and flood stage estimation. Since the webcams provided images with different resolutions, the images were resized with padding to a size of $512 \times 512 \times 3$ making them compatible with the models. For the segmentation model training, two additional datasets were provided by Kaggle Water Segmentation Dataset (see Liang et al., 2020) as well as Flood Segmentation Dataset (Pally et al., 2022) was used. Liang et al. (2020) and Pally et al. (2022) contain, respectively a total of 2,320 and >9,000 images of annotated segmentation for flooded areas and water bodies with a great variance in the features. Figure 1 shows the location of two USGS gauging stations along with their respective flood stage time series data distribution plots.

2.3. Image Data Normalization

The neural networks trained on numerical data are unlikely to forecast any values outside the numerical range of the training set. It is therefore recommended to normalize the data set and remove linear trends before training, as the forecasts may poorly perform otherwise. Normalization and trends removal also help reduce the sensitivity of networks to hyperparameters, reduce training time, and lead to forecasts that resemble the behavior of the historical data more faithfully. To improve convergence and reduce training times, the datasets of this research were normalized using Ioffe and Szegedy (2015) approach. Let μ be mean and σ the standard deviation of the training data set. Then the training and testing datasets were normalized by

$$z = \frac{x - \mu}{\delta} \tag{1}$$

Every data set uses a k-fold cross validation for splitting the training and testing data set (we set k to the standard value of 10). The k-fold cross-validation procedure is a standard approach for estimating the performance of configuration on a data set. The k-fold cross-validation procedure divides a limited data set into k non-overlapping folds. We used repeated k-fold cross-validation, by randomly shuffling the data, as a way to improve the performance. This involved repeating the cross-validation procedure multiple times and reporting the mean result

across all folds from all runs. This means the result is expected to be a more accurate estimate of the true unknown underlying mean performance of the model on the data set, as calculated using the standard error. Each of the k folds (10 folds here) was given an opportunity to be used as a held back test set, whilst all other folds collectively are used as a training data set. A total of k models was fit and evaluated on the k hold-out test sets and the mean performance was reported. 80% of the data was used for training and 20% for testing.

3. Methodology

3.1. Neural Networks Algorithms for Image Data

This section explains the workflow and algorithms that were developed for flood stage estimation using the USGS live web camera images. It should be noted that we used CNNs for both image data as well as time series data but with different setups and structures.

3.1.1. CNNs

CNNs are the most well-known neural networks for image segmentation. These algorithms showed enormous success in a variety of deep learning problems (LeCun et al., 2015; Pally & Samadi, 2022). CNNs are based on convolutional operations, which use filters iterating over the images to extract features from the pixels. Since the filters lay over multiple pixels, they are able to extract wider patterns and structures in the images (see Appendix A for the CNN structure). When filtering the size of images, a CNN is not only able to take into account surrounding pixels from images but also surrounding data (timewise) from a time series data set.

Details of the CNN workflow for flood stage prediction using image data are shown in Figure 2. As illustrated, the CNN model used for the USGS images was based on three 2D-convolutional layers with 32, 64, and 64 filters, respectively. Each filter has a size of 3×3, and the layers are connected with 2Dmax-pooling layers with a size of 2×2. After the convolutional layers, the model is flattened and followed by three-dense layers with 128, 64, and 1 output neurons for the regression task. All layers except the last one used the Rectified Linear Unit (ReLu) activation function as follows whereas the last dense layer uses the identity as its activation function (id(x) = x).

$$\mathrm{ReLu(x)} = \max(0, x) \tag{2}$$

3.1.2. Segmentation Model

We used a CNN (U-Net) for semantic segmentation of flood images driven from the USGS river cameras. Segmentation is an approach to identifying and labeling objects in images. In our study, a segmentation model is used to identify and label the waterbody of a given image. This approach drastically reduced the size of an image by a factor of 3 from $512 \times 512 \times 3$ to $512 \times 512 \times 1$. It also decreased features from the image, which were irrelevant for the flood stage estimation, and also de-noised flood images. For example, the waterbody was slightly invisible if the weather was cloudy or foggy. We used the U-Net approach proposed by Ronneberger et al. (2015), which is a convolutional network architecture for fast and precise segmentation of images using an encoder-decoder approach. The U-Net aims to efficiently generate accurate segmentation with minimum training data. The workflow that we developed to implement the segmentation model is illustrated in Figure 3. As a novel approach for the flood stage estimation, the U-Net is stacked in front of the CNN as a pre-processing step for image data with the goal of simplifying the task for the CNN implementation. As illustrated, the network's inputs are 572×572, 570×570, and 568×568, and the outputs are 392×392, 390×390, and 388×388. The network uses a skip connection to connect the upsampling result to the output of the submodule with the same resolution in the encoder as the input of the next submodule in the decoder.

3.2. Neural Networks Algorithms for Time Series Data

Multiple neural network algorithms were tested and compared to simulate the USGS flood stage data. The data set contains the flood stage of respective rivers and its corresponding time steps. The data starts on 1 January 2015 and ends on 2 September 2022. The data were formatted and preprocessed to remove trends using Ioffe and Szegedy (2015) approach explained in Section 2.3. We used the first 5 years of the >7-year calibration period as training data (1 January 2015 to 31 December 2019) while the last 2 years for validation and hyperparameter tuning (1 January 2020 to 9 February 2022). The first two subsets are used to derive the network's parametrization

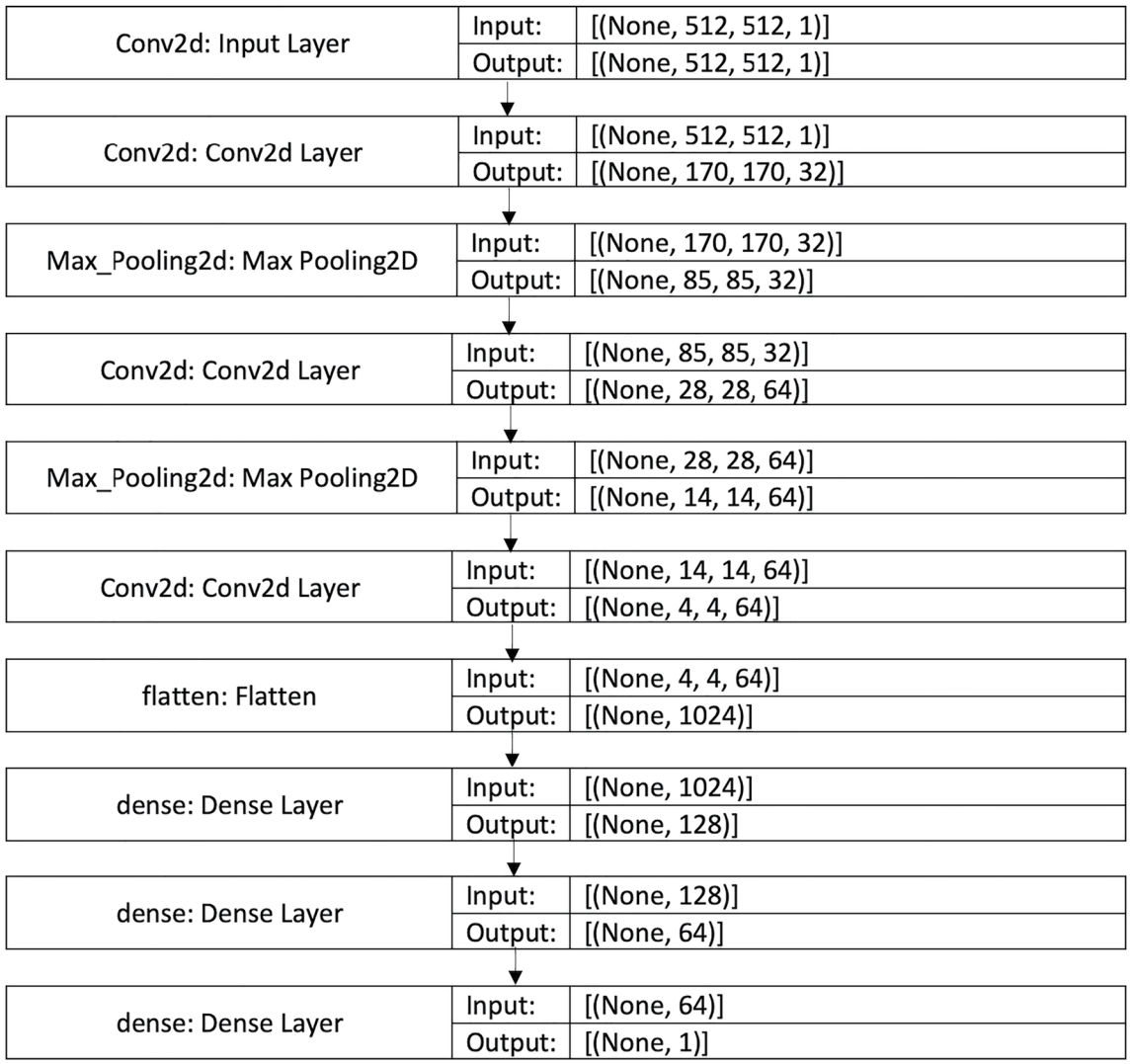


Figure 2. The workflow of Convolutional Neural Network architecture for the estimation of flood stage with an image size of $512 \times 512 \times 1$ and flood stage output.

(calibration in the context of hydrologic simulation) and the remainder of the data to diagnose the actual performance (validation in the context of hydrologic simulation).

We performed a trial-and-error process to tune the hyperparameters and derive the best values. The hyperparameters include sequence length of rainfall data map series (for the Columbus station only), batch size of 32, and the optimizer that compared with the learning curve. In the traditional hydrologic modeling approach, the number of iteration steps defines the total number of model evaluations performed during calibration (given an optimization algorithm without a convergence criterion; see Tabas & Samadi, 2022). The corresponding term for the neural network is the so-called epoch. One epoch is defined as the period in which each training sample is used once to update the model parameters (e.g., if the data set contains 1,000 training samples and the batch size is 10, one epoch would equal 100 iterations; the number of training samples divided by the number of samples per batch). In each iteration, 10 of the 1,000 samples were taken without replacement until all 1,000 samples were used once. This makes, each time-step of the flood stage data be simulated exactly once. This is similar to one iteration in traditional hydrologic model calibration, with a significant difference in generating every sample independently (see Kratzert et al., 2018).

The neural network models used for flood stage time series simulation are introduced in the following sections. To simplify the notation in this section, we denote x to be the number of time steps, which each model has the objective to predict in advance. We used a dynamic sliding window method on each output (see Girihagama

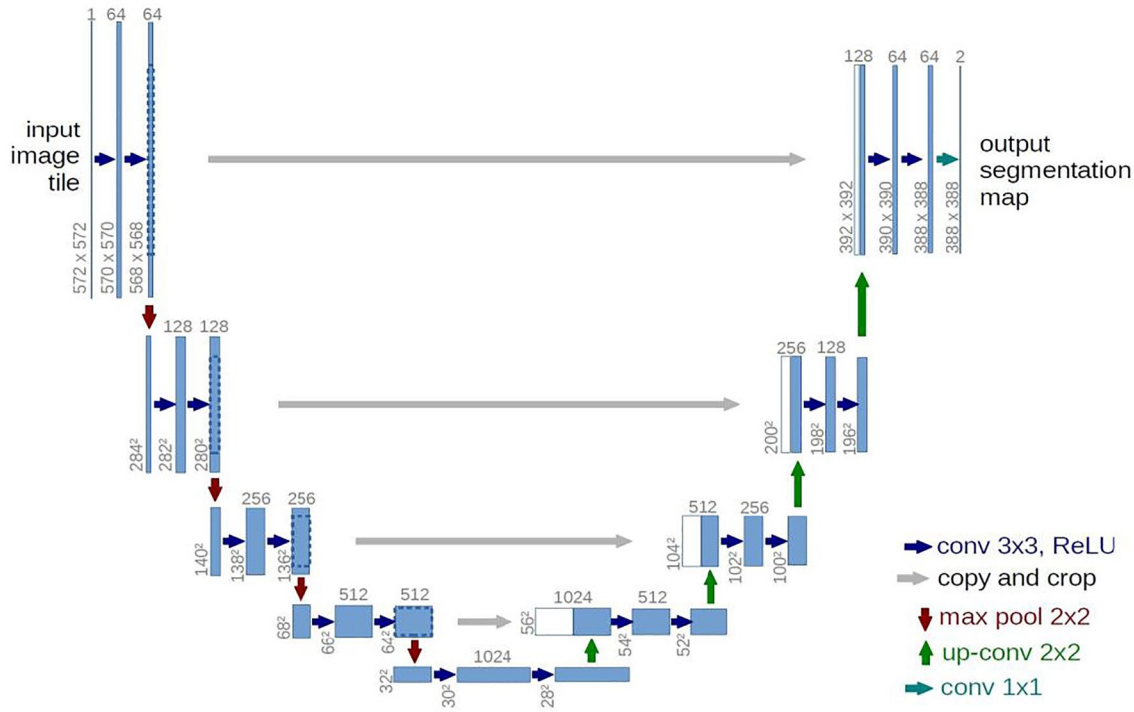


Figure 3. The architecture of U-Net with input as image and output as segmentation map was implemented in this study (partially adapted from Ronneberger et al. (2015)).

et al., 2022). When a neural network produces an output then this output was appended to the end of the input and the first value of the input is removed.

3.2.1. Dense Model

Our proposed dense model is an eight-layer stacked dense neural network connected by 128, 64, 32, and 4 neurons in the first four dense hidden layers and 1 neuron in the last output layer with the ReLu activation function (see Appendix A for the dense structure). After the third and the fourth dense hidden layers, the neural network contains dropouts with the rate of 0.5 which means half of the neurons in the previous layer would be ignored. Although, the dense model was tested using multiple dropouts lower than 0.5 to reduce independent learning among the neurons all values resulted in overfitting. ReLU is a piecewise linear function that outputs the input directly if it is positive, otherwise, it outputs zero. ReLU eases the training process and often achieves better performance. The derivative of ReLU is also easy to calculate when updating the weights of a node as part of the backpropagation error. Since the dense model only has a feed-forward connection, the network was unable to take previous data into account and only used the current events. In the background, the dense layer performed a matrix-vector multiplication. The values used in the matrix were some parameters that were trained and updated during the backpropagation process. The workflow of an eight layers deep dense model used in this study is shown in Figure 4.

3.2.2. CNNs

The next model that we used was a CNN that was trained on time series data. Due to the success of CNNs on image data, these networks were adopted and used for various simulation problems including time series data analysis, as well (Gamboa, 2017). The implemented model consists of a 1D convolutional layer with 64 filters as a first hidden layer and a kernel size of two using ReLu activation function. The second hidden layer consists of a 1D MaxPooling layer followed by a flattened layer in the third hidden layer. The output layer is a dense layer with one neuron. The CNN contained 64 filters with two kernels. The workflow of CNN implementation is shown in Figure 5.

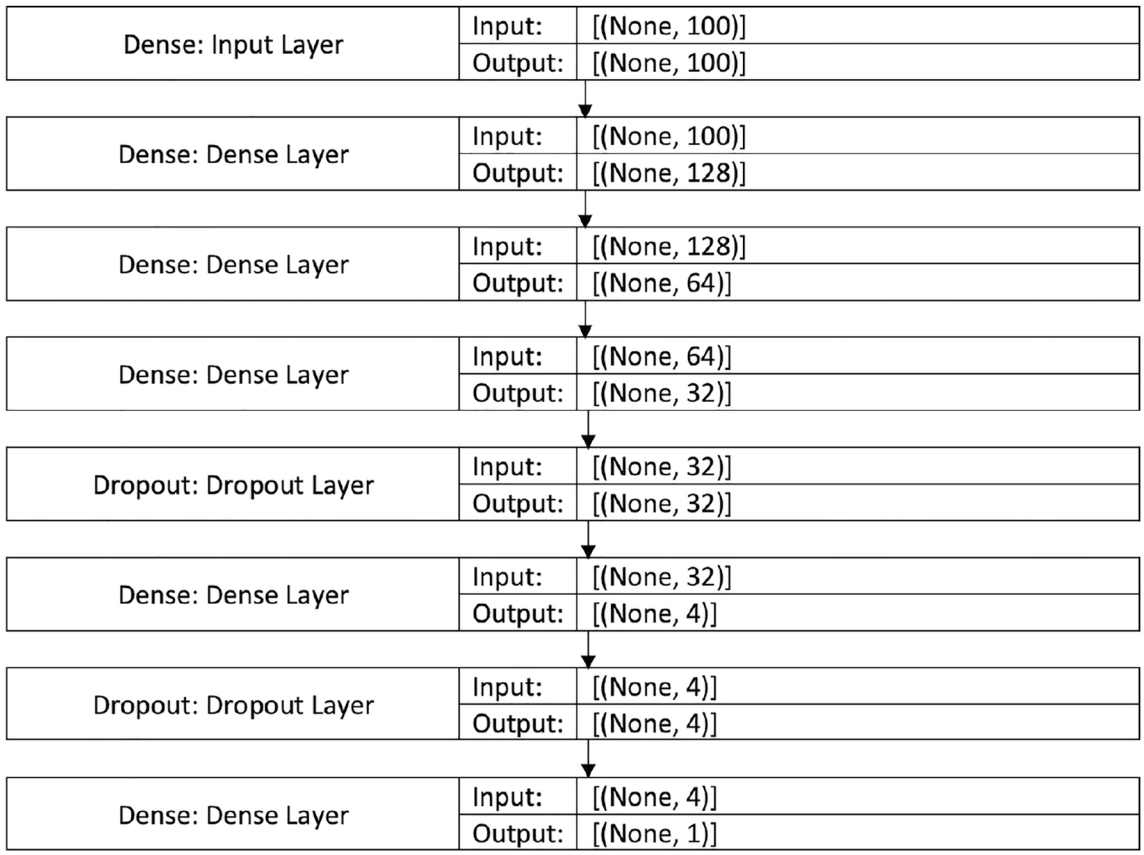


Figure 4. The architecture of a five layers deep dense model with historical gauge heights value input and predicted gauge height output for flood stage prediction.

3.2.3. LSTM

We used LSTM (developed by Hochreiter and Schmidhuber (1997)) which consists of multiple cells and each cell has a feedback connection to its neighbor cell, enabling the network to learn greater consecutive sequences (see Figure 6). The LSTM cell consists of a forget gate f_t, an input gate i_t and an output gate o_t and has a cell state c_t. At every time step t, the cell gets the data point x_t with the output of the previous cell h_{t-1}. The forget gate then defines if the information is removed from the cell state, while the input gate evaluates if the information should be added to the cell state and the output gate specifies which information from the cell state can be used for the next cells. We used four LSTM layers with 264, 132, 64, and 32 cells in the first four hidden layers, which were

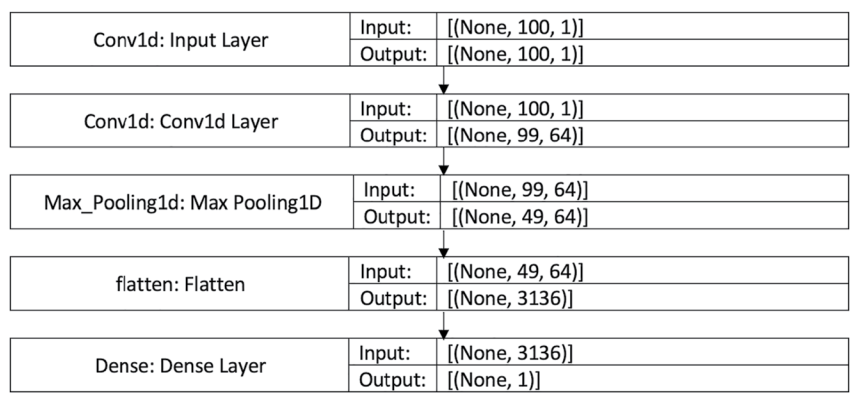


Figure 5. The architecture of Convolutional Neural Network with gauge heights values as input and predicted gauge height data as output.

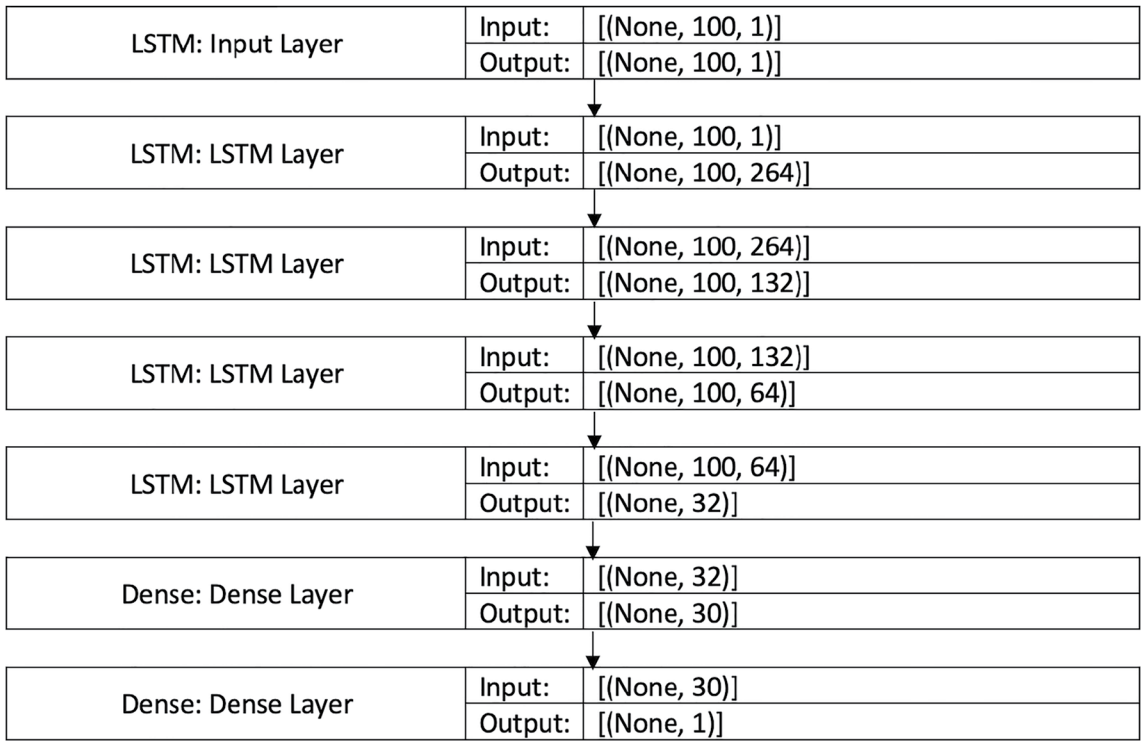


Figure 6. Proposed architecture for the Long Short-Term Memory (LSTM)-dense model, with gauge heights values as input and predicted gauge height data as output, used in this study for flood stage prediction. The model is a four-layer deep LSTM with a two-layer deep dense model.

then connected to two dense layers with 30 neurons followed by one neuron output layer (see Appendix A for the LSTM structure). The LSTM simulation was performed with these input layers along with Adam optimizer, ReLu activation, and a single lagged dependent-variable value to train with a learning rate of 0.00001 and no dropout. The architecture of our proposed LSTM-dense model is illustrated in Figure 6.

3.3. Performance Metrics

Several key performance metrics are used to evaluate the performance of the neural networks. For image data performance, we used the Sørensen-Dice coefficient (Sorensen, 1948) which is one of the most well-known statistical measurements for evaluating the similarity between the true segmentation y and the predicted segmentation $\hat{y}$ of an image. We defined y as the ground truth and $\hat{y}$ *as* the predicted value by the models. We also defined $\bar{y}$ as the mean of ground truth value. The Dice loss is used for the training of the U-Net that is defined as follows:

$$\text{DICE} = 1 - \frac{2\,|y \cup \hat{y}|}{|y| + |\hat{y}|} \tag{3}$$

y_i is the ground truth value at a given time i and similarly $\hat{y}_i$ is the predicted value at a given time i. Since both the gauge height estimations of an image and the future flood stage predictions are regression problems, the same loss function and metrics were used for performance assessment. So, MSE (Equation 4) was used as an objective function or loss function for all models.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

For further performance assessment, the MAE, RMSE, MAPE, and the Weighted Average Percentage Error (WAPE) were also used as additional metrics that are defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5}$$

$$\text{RMSE} = \frac{\sqrt{1}}{\text{MSE}} \tag{6}$$

$$\text{MAPE} = 100\,\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

MAPE is the most used relative percentage error metric that was used in this research to compare different data sets.

$$\lim_{y\to 0}\text{MAPE} = \lim_{y\to 0} 100\,\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{8}$$

The WAPE was used as a sum in the denominator to improve numerical stability while still being a relative percentage error metric.

$$\text{WAPE} = 100\,\frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i|} \tag{9}$$

We used MSE, MAE, RMSE, and MAPE for image data performance assessment while the Nash-Sutcliffe efficiency (NSE; Equation 10, Nash & Sutcliffe, 1970) was also used for time series performance assessment.

$$\text{NSE} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)} \tag{10}$$

4. Results

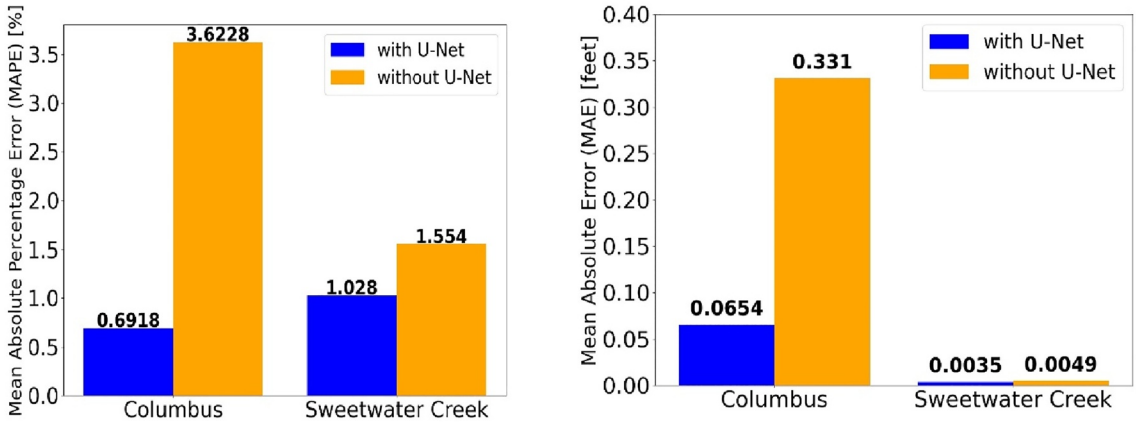
The experiments were benchmarked on a server running Arch Linux with 256 GB RAM and an NVIDIA Tesla P100 GPU with 16 GB Memory. The code is implemented in Python 3.9 using TensorFlow 2.6. We first present flood stage simulation using USGS images and then discuss flood stage time series prediction. This is followed by the analysis of the results, demonstrated in the following sections.

4.1. Flood Stage Estimation Using USGS Images

The CNN was first implemented with and without the U-Net as a pre-processing step to evaluate its robustness in flood stage estimation. The CNNs with and without U-Net were trained up to 150 and 100 epochs, respectively. After the training, the U-Net generated a precise segmentation of the body of the water. This removed unnecessary features like the riverbank's environment, colors, noise, etc., from the images as they were not important for the water depth estimation.

As noted in Figure 7, the relative error decreased from 3.6% to 0.69% at the Columbus station which is a relative improvement of $1 - \frac{0.6918}{3.6228} \approx 80.9\%$. The same result was observed with the MAE reduction from 0.33 to 0.065 feet using the U-Net for the pre-processing. This is a similar improvement of $1 - \frac{0.0654}{0.331} \approx 80.2\%$. At the Sweetwater Creek station where we observed a minimal absolute reduction due to a low average gauge height, the impact of the U-Net still exists with a relative improvement of $1 - \frac{0.0035}{0.0049} \approx 28.6\%$ for the MAE and $1 - \frac{1.028}{1.554} \approx 33.84\%$ for the MAPE metrics. Results of the U-Net simulation are illustrated in Figure 8. Figure 8c was streamed directly from the USGS river cams which looks a little foggy (low resolution) and could add noise to the simulation. However, the U-Net was able to dismiss and ignore unnecessary features in the images (see Figure 8d for example) and estimate the segmentation area. This approach improved the performance of the CNN by requiring fewer data and speeding up the convergence of the model (Figure 9).

Table 1 summarizes the modeling performances on the testing data set. These results reveal that CNN has higher accuracy if the U-Net is stacked in front of the network. The absolute error with the U-Net is 0.0654 feet ≈2 cm at the Columbus Station while it is 0.0035 feet ≈0.108 cm at the Sweetwater Creek station, which are practically low errors for flood stage estimations.



(a) MAPE values with and without the U-Net. (b) MAE values with and without the U-Net.

Figure 7. Comparison of performance metrics for flood stage estimation with and without the U-Net at the Sweetwater Creek (left) and the Columbus (right) gauging stations. (a) Mean absolute percentage error values with and without the U-Net. (b) Mean absolute error values with and without the U-Net.

4.2. Flood Stage Prediction Using Time Series Data

LSTM, CNN, and dense models were implemented to simulate historical flood stage data from 1 January 2015 to 2 September 2022 for the Columbus and Sweetwater stations. We used precipitation data as an input variable for the Columbus simulation while the Sweetwater flood stage simulation was performed using flood stage data only. The pre-trained models were then used to forecast flood stage data in advance for multiple timescales (i.e., 6, 12, 24, and 48 hr). Each forecast had access to the last 100 data points as its time-series data set, which corresponds to the last $\frac{100 \times 15}{60} = 25$ hr. Each model was trained using 50 epochs. A complete overview of all metrics for each experiment during the training period can be found in Table 2. As shown, the LSTM algorithm performed the best for both stations, although Sweetwater Creek simulation is slightly better than Columbus. LSTM has special inner gates that allow the algorithm to learn which data in each sequence should be kept and correspondingly which data can be forgotten (remembering information for a longer time) or ignored. This led to consistently better performance in time series prediction. Furthermore, the structure of LSTM makes it an excellent network for solving high dimensionality problems in data prediction because it enables gradients to flow through time readily. It is interesting to note that most networks showed deficiencies in predicting low flood stage values. This might be related to the fact that like conceptual hydrologic models, neural networks don't deal with the physical processes (hydraulic conductivity, soil storage capacity, etc.) of the catchment system. It seems that computing low flood stage values when a shallow aquifer/groundwater system is the primary contributor to river gauge data (see Samadi et al., 2018) is particularly difficult for neural network models (Figure 10).

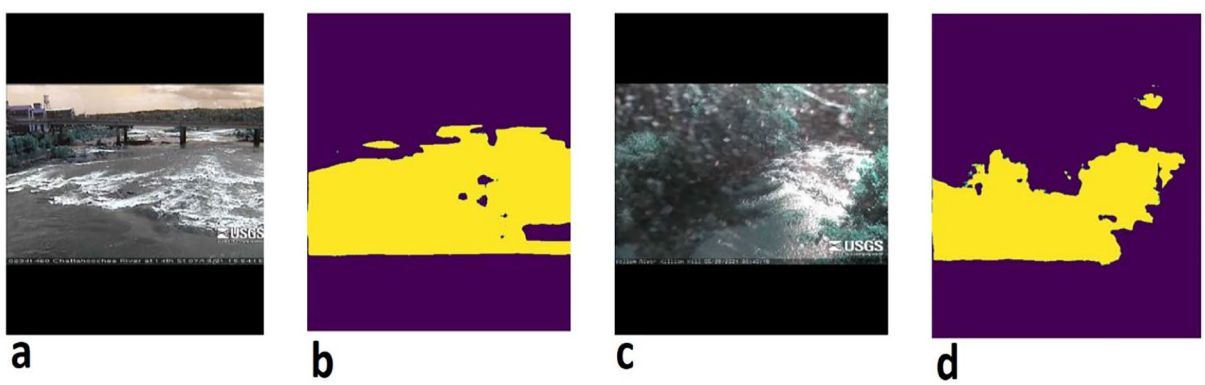


Figure 8. U-Net segmentation results on example images. (a) Convolutional Neural Network (CNN) input data of the Columbus station without the U-Net. (b) CNN input data of the Columbus station with the U-Net as a pre-processing step. (c) CNN input data of the Sweetwater Creek with noise in front of the camera. (d) U-Net result of the Sweetwater Creek that was able to remove the noise in front of the camera.

10.1029/2022EA002385

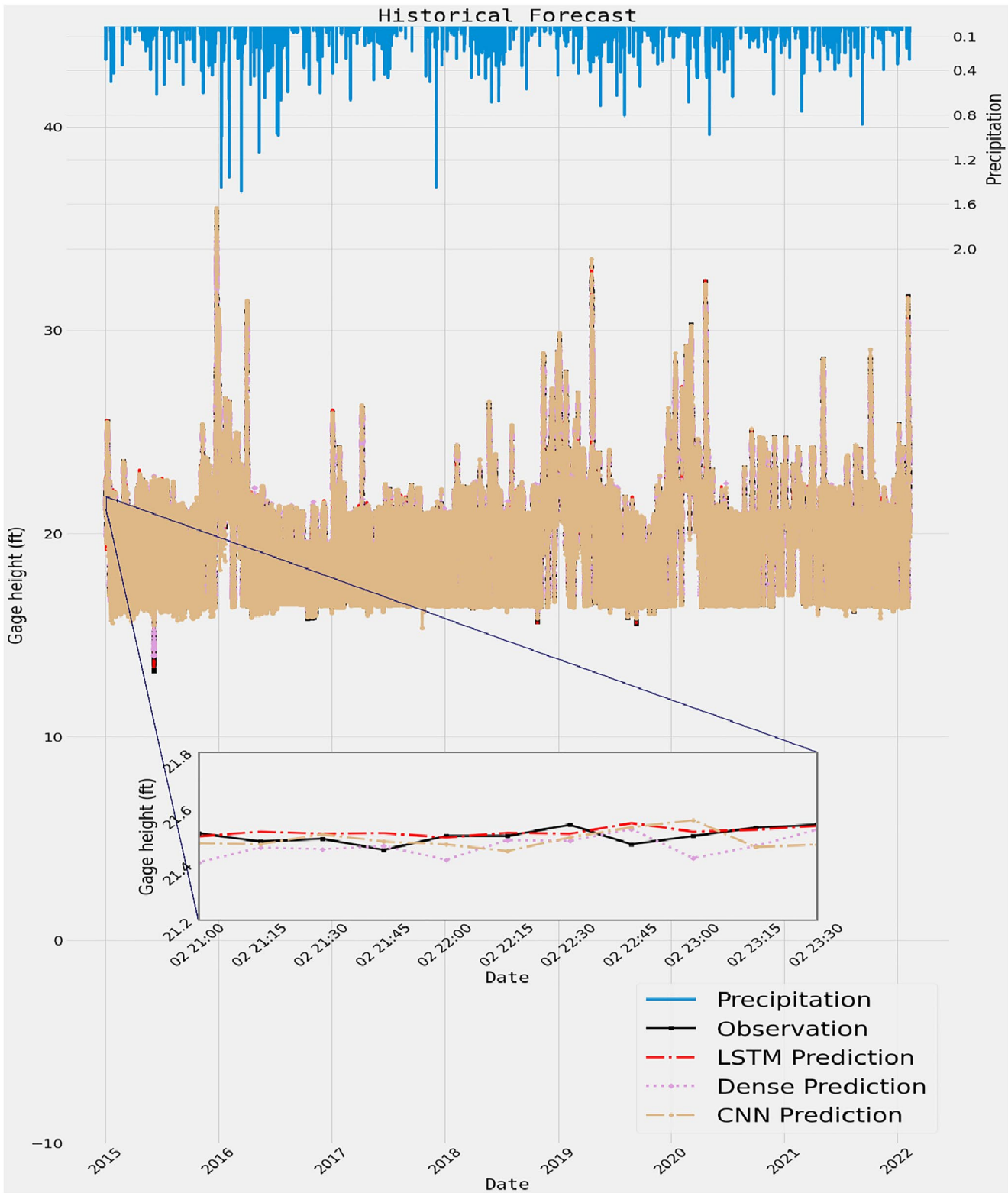


Figure 9. Historical flood stage simulation using the Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and dense models for the Columbus station. Rainfall data was used to simulate flood stage at this station only.

Table 1
Flood Stage Estimation Results Using the Testing Data Set for the Two US Geological Survey Gauging Stations

	Performance results with U-Net				Performance results without U-Net			
Station name	MSE	MAE	RMSE	MAPE	MSE	MAE	RMSE	MAPE
Columbus	0.0066	0.0654	0.0814	0.342	0.2533	0.331	0.5033	1.7537
Sweetwater	0.000198	0.0035	0.044	0.2634	0.00038	0.0049	0.0062	0.3807

Note. Units are in feet.

Flood stage forecasts for different timescales are presented in Table 3. Among all models, the LSTM was more proficient in forecasting flood stage, particularly for longer periods (24 and 48 hr) with respect to the WAPE index. For the LSTM with a 6 hr lead time, the NSE was 85% for the Columbus gauging station whereas it was 83% for the Sweetwater Creek. For the 12 hr forecast, the LSTM model at the Columbus station achieved an NSE of 96% and a WAPE around 0.25. For a 24 hr forecast, however, the LSTM performance remained consistent with an NSE of ≈96% at the Columbus gauging station. Again, for the 48 hr forecast, the LSTM performance remained consistent with an NSE of ≈95%. The LSTM performance at the Columbus station is comparable with a case study of flood stage forecast in Italy (see Dazzi et al., 2021). Unlike Dazzi et al. (2021), we incorporated weather features and precipitation data into the networks during the training period for this station which improved the simulation results significantly.

Additionally, all models performed better and remained consistent for long lead time forecasts (12, 24, and 48 hr) than short-term forecasts. This is probably due to the fact that the neural networks learned the long-term fluctuations in flood stage data more precisely compared to short-term fluctuations. This may also reveal the fact that the models learned the long-term seasonality and trends in the data fairly well. The dense and CNN performed a few negative NSE values. This is because the error variance estimation of both models was significantly larger than the variance of the observations. Among all models, CNN performed more negative NSE values because this network learns spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, and pooling layers while it fails to consider temporal variability or sequences in the data. Overall, the CNN model provided very good results for 12 and 24 hr as well as 48 hr forecasts, however, its performance slightly diminished during shorter forecasts. Our designed CNN contained a convolutional layer with 64 filters in the first layer and a pooling layer where we used max pooling in the second layer. In the max pooling layer, N number of data was maxed into a single data and then scaled down the number of available data. The pooling layer, as it scaled the data down, showed great advantages in the image data set, although this approach revealed some deficiencies and drawbacks for time-series simulation. Finally, the output layer contained one neuron. The dense model, on the other hand, was constructed using 128 cells in the first layer, 64 cells in the second layer, 32 cells in the third layer, 4 cells in the fourth layer, and lasting 1 cell in the fifth layer which was the output layer of the network. The dense layer also contained 1 dropout in the third and the fourth layers. These dropouts varied considerably between 0.2 and 0.5 which means the third and the fourth layers of the dense neural network ignored 20%–50% of the neurons in the cells. This approach prevented all the neurons in the third and the fourth layers from synchronously optimizing their weights. As we monitored the dense network performance, we found that ignoring some of the neurons had no degradation on the performance.

Table 2
Performance Metrics for All Experiments During the Training Period

Station name	MAE	MAPE	WAPE
Dense		Performance metrics	
Columbus	0.0225	10.2111	8.1885
Sweetwater creek	0.0312	5974.6763	15.7255
CNN		Performance metrics	
Columbus	0.0065	651.2095	2.3637
Sweetwater creek	0.0551	1.9092	1.9974
LSTM		Performance metrics	
Columbus	**0.0022**	**15.4133**	**0.8065**
Sweetwater creek	**0.0017**	**0.9897**	**0.8535**

Note. The mean absolute error (MAE) values are normalized as described in Section 2.3. Best modeling performances are exhibited in bold.

One can also note the difficulties of the models in flood stage simulation at the Sweetwater Creek station. The modeling results at the Sweetwater Creek station appear to be less proficient compared to the Columbus station. One possible explanation is that Sweetwater Creek is a tributary of the Chattahoochee River and significantly a larger river. This concludes a major dependency on the Chattahoochee River and makes the forecast more challenging due to additional influences on gauge height values from non-supervised features such as shallow aquifer system contribution to the river system, extensive alluvial and non-alluvial forested wetlands, and wide floodplains

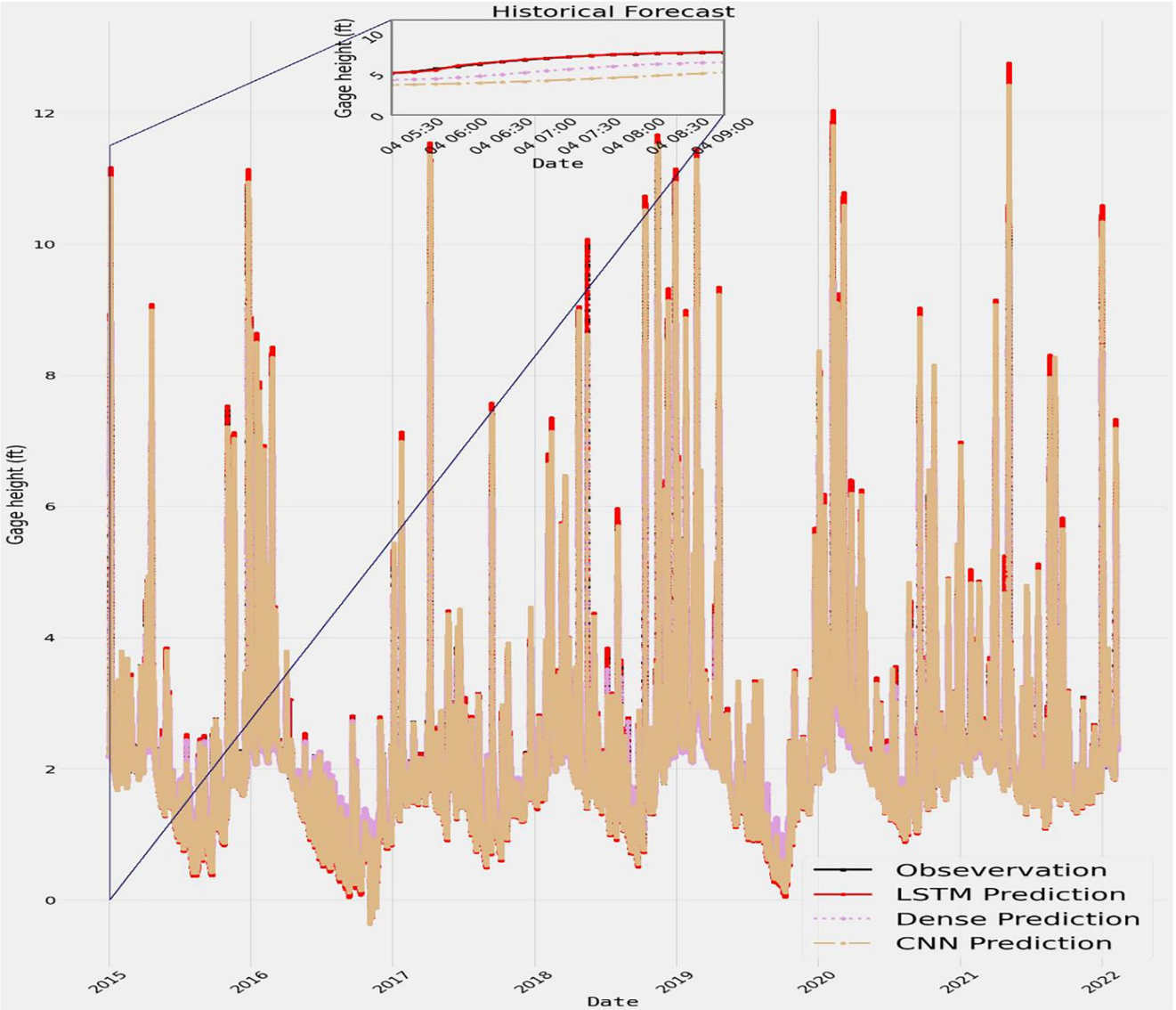


Figure 10. Historical flood stage simulation using the Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and dense models for the Sweetwater Creek station.

as stated elsewhere (see Amatya & Jha, 2011; Samadi et al., 2018). These non-supervised features also cause substantial fluctuations (noise) in gauge height values in which all deep neural network models were tempered to learn by this heterogeneity, particularly over the long-term forecasts.

Prediction results on two example batches of datasets are visualized in Figures 11 and 12. As shown in Figure 11, the LSTM was able to accurately predict a quick rise and fall of the flood stage data (quick noise and fluctuation) at the Columbus station, while the peak and minimum data were not predicted correctly. Although the slope and behavior of the curves are well predicted for rapid-changing flood stage data. The other models either predicted only a rise of the flood stage (the first batch) or predicted no changes at all (the second and the third batches).

For the Sweetwater Creek station (see Figure 12), all models correctly predicted a continuous decrease in the gauge level. The LSTM provided great results for the first and the second baches while for the third batch none of the models showed good prediction. The CNN also performed well on the first batch and predicted the rise and fall of flood stage data well. As shown, the LSTM revealed a great advantage over the other models, especially at the Columbus station where the flood stage fluctuates were more frequent across a 12-hr lead time. This might be related to incorporating weather features (precipitation data) into the network in this station which made the

Table 3
Flood Stage Forecasting Performances Across Short to Long Lead Times for Each Algorithm

Station name	MSE	MAE	NSE	WAPE	MSE	MAE	NSE	WAPE
Dense model	Performance results for 6 hr forecasting				Performance results for 12 hr forecasting			
Columbus	0.0354	0.1585	0.6022	0.6607	0.0152	0.1066	0.9434	0.4547
Sweetwater	0.0127	0.1068	−0.6614	3.3465	0.0074	0.0728	0.5745	2.3641
Dense Model	Performance results for 24 hr forecasting				Performance results for 48 hr forecasting			
Columbus	0.0177	0.1209	0.8938	0.5116	0.0227	0.1328	0.8768	0.5584
Sweetwater	0.0074	0.0728	0.5745	2.3644	0.0105	0.0923	0.7879	3.3427
CNN Model	Performance results for 6 hr forecasting				Performance results for 12 hr forecasting			
Columbus	0.0503	0.1052	0.4350	0.4388	0.0142	0.0711	0.9472	0.3032
Sweetwater	0.0450	0.1839	−4.8746	5.7578	0.0237	0.1126	−0.3478	3.6536
CNN Model	Performance results for 24 hr forecasting				Performance results for 48 hr forecasting			
Columbus	0.0080	0.0530	0.9520	0.2243	0.0261	0.0865	0.8585	0.3637
Sweetwater	0.0237	0.1126	−0.3478	3.6536	0.0071	0.0551	0.8557	1.9974
LSTM	Performance results for 6 hr forecasting				Performance results for 12 hr forecasting			
Columbus	**0.012**	**0.0646**	**0.8541**	**0.2694**	**0.0091**	**0.0597**	**0.9662**	**0.2547**
Sweetwater	**0.0012**	**0.0271**	**0.8349**	**0.8489**	**0.0022**	**0.0368**	**0.8693**	**1.1950**
LSTM	Performance results for 24 hr forecasting				Performance results for 48 hr forecasting			
Columbus	**0.0055**	**0.0469**	**0.9670**	**0.1987**	**0.0087**	**0.0586**	**0.9529**	**0.2466**
Sweetwater	**0.0036**	**0.0469**	**0.8809**	**1.5967**	**0.0035**	**0.0441**	**0.9286**	**1.6002**

Note. The best performances are shown in bold.

patterns in flood stage data more learnable for the network. Among all models, only the LSTM was able to predict the slope and patterns of the flood stage while the other two models struggled to capture such complex behavior.

5. Discussion and Conclusions

This research consisted of two different systems that were integrated together to create an end-to-end automated flood detection and gauge height prediction system. The proposed system was designed and developed to accurately estimate and predict flood stage values using image data driven in real-time from the USGS river web cameras as well as time series data. We selected two USGS gauging stations where river cam images and flood stage data were available for validation at both stations. We first predicted the historical period (starting on 1 January 2015 and ending on 2 September 2022) and then we used the pre-train models for near real-time flood stage forecast. The pre-trained models were able to skillfully forecast flood stage up to 48 hr ahead of time. To perform the image-based prediction, the system used a novel approach of a CNN stacked on top of a segmentation model as a pre-processing step to reduce unnecessary features and noise from the images. Using this approach, the performance and the accuracy were relatively improved up to 80% (MAPE of 0.69% vs. 3.62%). This provided more reliable flood stage estimation using live images and removed the burden of using a large number of images for a precise prediction. The results on the testing datasets were accurate enough for any practical application. Compared to the dense model and the CNN, the LSTM algorithm, despite requiring the longest training time, quickly forecasted near real-time gauge heights values in a fraction of time (a few seconds). For the Columbus station, the LSTM forecasted historical gauge height data with an NSE of 85% for 6 hr, 96% for 12 hr, 96% for 24 hr, and 95% for 48 hr. This network also provided NSE values of 83%–92% respectively, for shorter to longer-term predictions at the Sweetwater Creek station. Among all models, the training time of CNN was lengthy (>4 hr) that required a large recognition computing; however, the CNN network performed the testing task relatively in a short amount of time ranging from a few seconds to 3 min.

Other studies received similar accuracy, for example, a relative error of 5% for 9 hr forecast was reported by Dazzi et al. (2021), however, they used catchment physical features in the model in addition to the discharge data

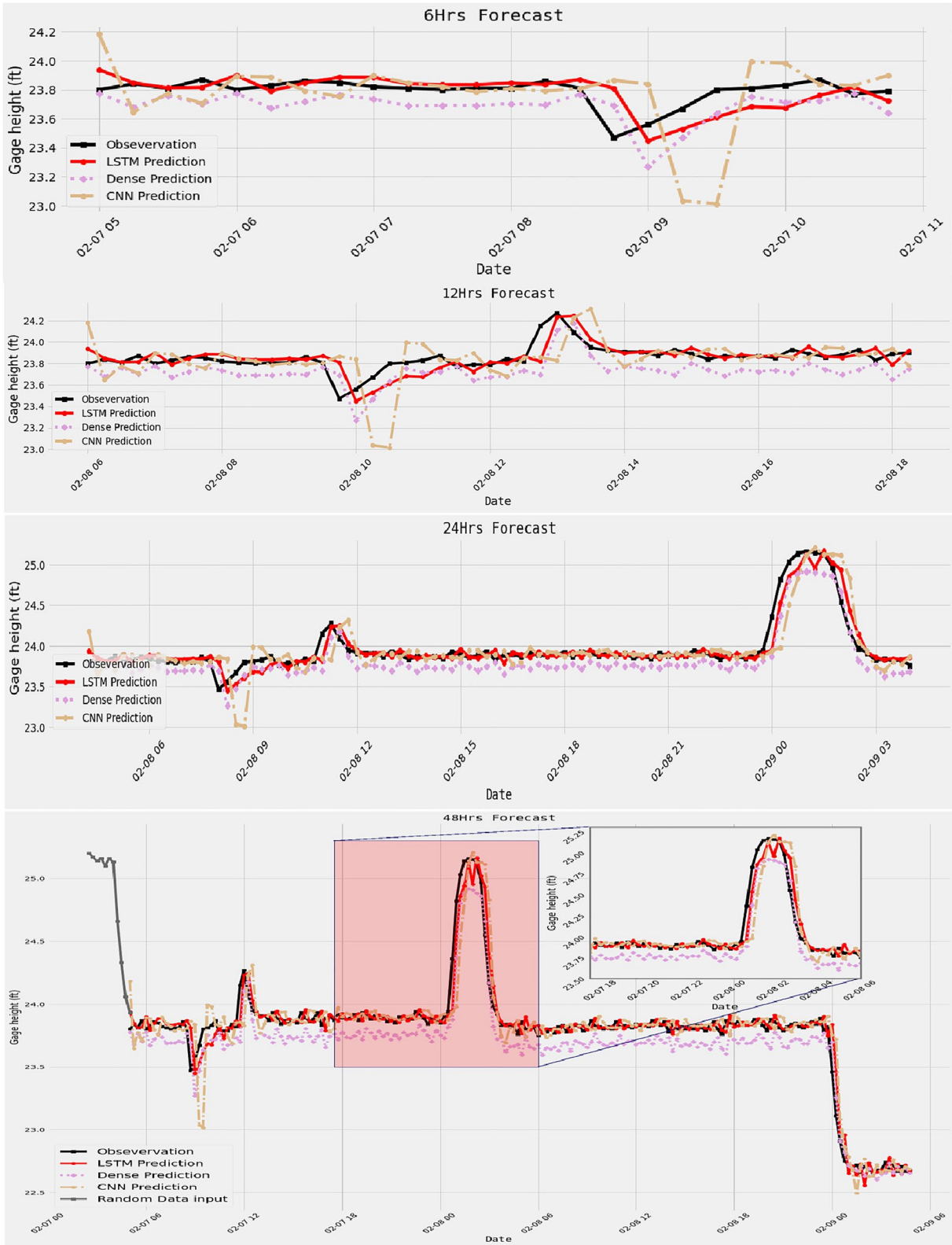


Figure 11. Flood stage forecasting for different time scales ranging from 6, 12, 24 to 48 hr in the Columbus station.

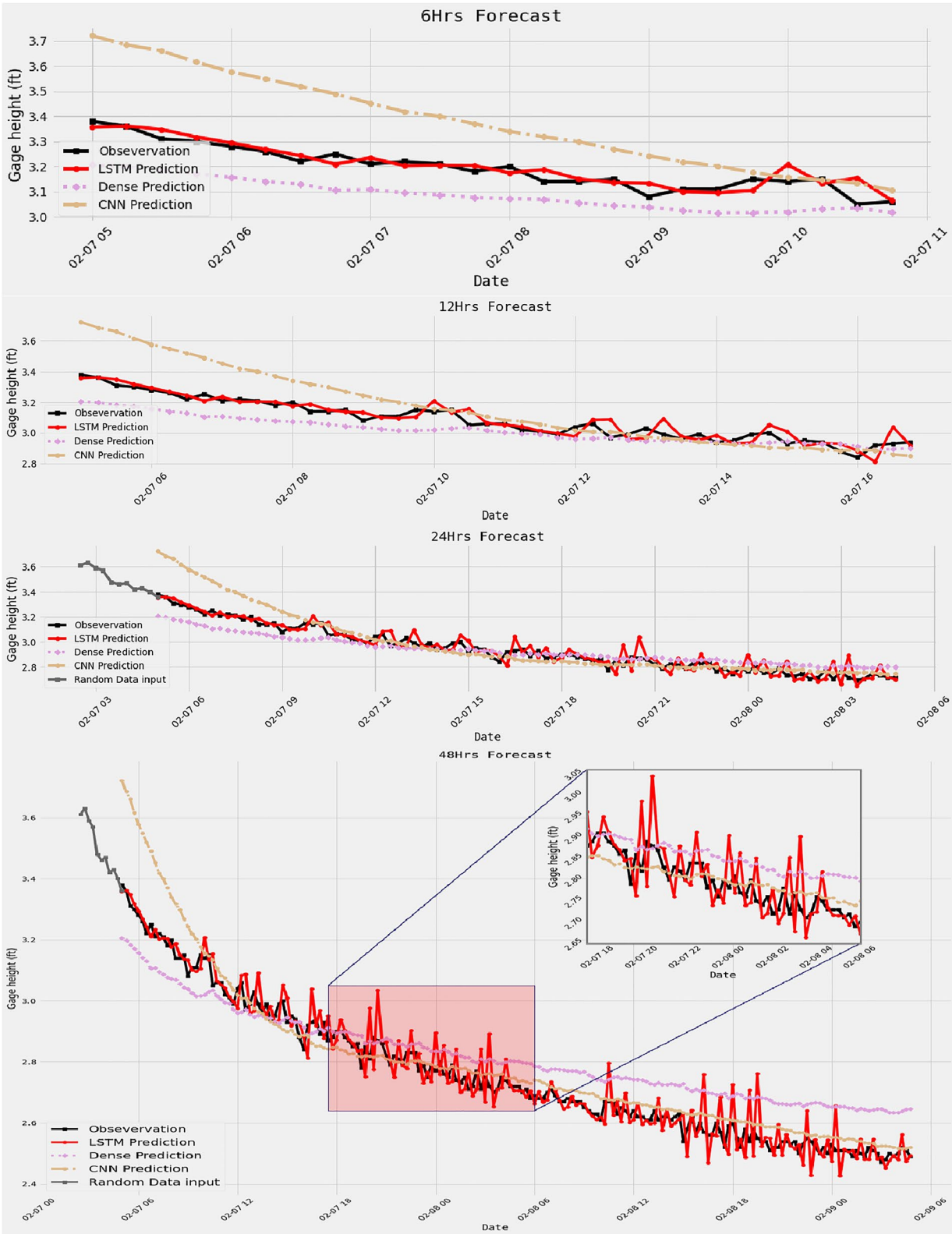


Figure 12. Flood stage forecasting for different time scales ranging from 6, 12, 24 to 48 hr in the Sweetwater Creek station.

which helped improve prediction results across the larger catchment. Since the CNN and the LSTM were the most robust and accurate models in predicting flood stage data, these networks should be preferred for setting up an operational forecasting system using both images and time series data. These two algorithms can be embedded in the USGS live camera as a real-time image-time series data-based early warning system. LSTM results can also be integrated with the current flood forecasting model such as the National Water Model (NWM) to complement and speed up real-time forecasts (a hybrid NWM-LSTM emulator) in any region where the timely and accurate warning is vital for flood emergency management and response.

This study only used the precipitation data as an input feature for the Columbus station that was aggregated for different time scales (6, 12, 24, and 48 hr). Our results showed that incorporating rainfall features are adequate for small to medium lead time forecasts while longer lead time requires more physics-based features to be incorporated into the LSTM architecture, particularly for large-scale catchment systems. Since deep neural networks do not deal with physical processes (shallow aquifer parametrization, complexities in floodplain storage capacity, flow resistance factor, hydraulic conductivity, etc.) of the catchment system, it appears forecasting flood stage data across large catchments is especially challenging for these algorithms. Deep neural networks are only as good as the training, and it is much harder to model a "proxy-truth" to train networks on. One possible way to improve deep neural network forecasting would be using a proper transform scheme such as the Fourier transform that can transform a data signal from its time domain to its frequency domain. The peak values in the frequency spectrum indicate the most occurring frequencies in the signal. The larger and sharper a peak is, the more prevalent a frequency is in a signal. The location (frequency-value) and height (amplitude) of the peaks (extreme gauge height values here) in the frequency spectrum can then be used as input (learnable weights) for deep neural networks training.

As the result of this study, we also included images from the Columbus and the Sweetwater Rivers to Flood Segmentation Dataset (Pally & Samadi, 2022) with their associated annotated labels. We hope this data set will encourage good practice and provide a foundation for others to build on. To improve the use of image data in hydrological simulation, more research is still needed to understand the accuracy of image-based estimation. Furthermore, poor resolution images can increase errors and uncertainty in simulation. Some image enhancement techniques such as power-law and logarithmic transformation (Maini & Aggarwal, 2010) can be particularly helpful to deal with the issue of poor lighting conditions in an image. Other techniques such as histogram equalization (Wang et al., 1999) and image thresholding transformation (Perez-Sanz et al., 2017) can also be effective in improving the resolution of live images. These methods are particularly useful in image segmentation to isolate an image of interest from the background. Although, based on the type of image and type of noise with which it is corrupted, a combination of image enhancement methods can be used to improve visual quality.

To summarize, the presented results are promising for both time series data as well as image applications. As we make progress in neural network simulation, we expect to gain a new understanding of the relative importance of the choice of neural network architecture, hyperparameter tuning, and parameter values for different flood stage case studies, a new understanding of how to capture short-term, extreme fluctuations in the data, and a new understanding of the limitations of neural networks in effectively discriminating among competing modeling structures. The analyses presented herein are intended to provide a basis for both image and time series data applications in flood stage prediction across complex environmental settings. However, subsequent in-depth studies are needed to examine the individual algorithms in different case studies and check emergent behavior of the network (in more detail) over time using carefully designed experiments to expose key network parameters and limitations, and accounting for error and uncertainty in both models forcing and input data. Acknowledging a growing enthusiasm for neural network modeling in hydrology community, we expect progress on multiple fronts: a better hybrid model to simulate/forecast flood stage using both image and time series data, better benchmarking and more accuracy in short term forecast, better accuracy metrics, and better preprocessing, error estimation, and noise reduction approaches. As always, we invite dialogue with water resources communities interested in this and other related deep learning applications for flood simulation problems.

Appendix A: A Summary of TensorFlow Parameter Values for Gauge Height Prediction Models With the Number of Arguments

Dense Model's parameter values, and trianable parameters are displayed below.

Dense Model

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	12928
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 32)	2080
dropout (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 4)	132
dropout_1 (Dropout)	(None, 4)	0
dense_4 (Dense)	(None, 1)	5

Total params: 23,401
Trainable params: 23,401
Non-trainable params: 0

CNN's structure, parameter values, and trianable parameters are displayed below.

CNN

Layer (type)	Output Shape	Param #
conv1d_8 (Conv1D)	(None, 99, 64)	192
max_pooling1d_8 (MaxPooling 1D)	(None, 49, 64)	0
flatten_8 (Flatten)	(None, 3136)	0
dense_8 (Dense)	(None, 1)	3137

Total params: 3,329
Trainable params: 3,329
Non-trainable params: 0

LSTM's structure, parameter values, and trianable parameters. are displayed below.

LSTM

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 264)          280896

 lstm_1 (LSTM)               (None, 100, 132)          209616

 lstm_2 (LSTM)               (None, 100, 64)           50432

 lstm_3 (LSTM)               (None, 32)                12416

 dense (Dense)               (None, 30)                990

 dense_1 (Dense)             (None, 1)                 31

=================================================================
Total params: 554,381
Trainable params: 554,381
Non-trainable params: 0
_________________________________________________________________
```

Data Availability Statement

The USGS is acknowledged for providing free of charge live river web camera photos as well as timeseries data. The USGS 02337000 SWEETWATER CREEK NEAR AUSTELL, GA data is available at https://waterdata.usgs.gov/nwis/inventory/?site_no=02337000 [dataset]. The USGS 02341460 CHATTAHOOCHEE RIVER AT 14TH ST, AT COLUMBUS, GA data is available at https://waterdata.usgs.gov/nwis/inventory/?site_no=02341460 [dataset]. The code is available on GitHub [software] https://github.com/HHRClemson/Flood-stAge-predIction-thRough-Deep-Neural-Networks (Windheuser et al., 2022). The image data are available on Kaggle https://www.kaggle.com/datasets/hhrclemson/flooding-image-dataset [dataset].

Acknowledgments
This research was supported by the US National Science Foundation Directorate of Engineering (Grant CBET 1901646 and CMMI 2125283). Clemson University (USA) and Technical University of Munich (Germany) are acknowledged for generous allotment of computing time on the Palmetto cluster and Scyper Cloud, respectively.

References

Al-Fawa'Reh, M., Hawamdeh, A., Alrawashdeh, R., & Jafar, M. T. (2021). Intelligent methods for flood forecasting in Wadi al Wala, Jordan. In *In 2021 International Congress Of Advanced Technology and Engineering (ICOTEN)* (pp. 1–9). IEEE.

Amatya, K. M., & Jha, M. K. (2011). Evaluating the SWAT model for a low-gradient forested watershed in coastal South Carolina. *Transactions of the American Society of Agricultural and Biological Engineers*, *54*(6), 2151–2163. https://doi.org/10.13031/2013.40671

Baek, S.-S., Pyo, J., & Chun, J. A. (2020). Prediction of water level and water quality using a CNN-LSTM combined deep learning approach. *Water*, *12*(12), 3399. https://doi.org/10.3390/w12123399

Bermúdez, M., Ntegeka, V., Wolfs, V., & Willems, P. (2018). Development and comparison of two fast surrogate models for urban pluvial flood simulations. *Water Resources Management*, *32*(8), 2801–2815. https://doi.org/10.1007/s11269-018-1959-8

Bhola, P. K., Nair, B. B., Leandro, J., Rao, S. N., & Disse, M. (2019). Flood inundation forecasts using validation data generated with the assistance of computer vision. *Journal of Hydroinformatics*, *21*(2), 240–256. https://doi.org/10.2166/hydro.2018.044

Dazzi, S., Vacondio, R., & Mignosa, P. (2021). Flood stage forecasting using machine-learning methods: A case study on the Parma River (Italy). *Water*, *13*(12), 1612. https://doi.org/10.3390/w13121612

Donratanapat, N., Samadi, S., Vidal, M. J., & Sadeghi Tabas, S. (2020). A national-scale big data prototype for real-time flood emergency response and management. *Environmental Modelling and Software*, *10*, 104828. https://doi.org/10.1016/j.envsoft.2020.104828

Faruq, A., Arsa, H. P., Hussein, S. F. M., Razali, C. M. C., Marto, A., & Abdullah, S. S. (2020). Deep learning based forecast and warning of floods in Klang River, Malaysia. *Ingénierie des Systèmes d'Information*, *25*(3), 365–370.

Field, C. B., Barros, V., Stocker, T. F., & Dahe, Q. (2012). *Managing the risks of extreme events and disasters to advance climate change adaptation: Special report of the intergovernmental panel on climate change*. Cambridge University Press.

Gamboa, J. C. B. (2017). Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887.

Girihagama, L., Naveed Khaliq, M., Lamontagne, P., Perdikaris, J., Roy, R., Sushama, L., & Elshorbagy, A. (2022). Streamflow modelling and forecasting for Canadian watersheds using LSTM networks with attention mechanism. *Neural Computing & Applications*, *34*(22), 19995–20015. https://doi.org/10.1007/s00521-022-07523-8

Hilldale, R. C., & Raff, D. (2008). Assessing the ability of airborne LiDAR to map river bathymetry. *Earth Surface Processes and Landforms*, *33*(5), 773–783. https://doi.org/10.1002/esp.1575

Hinkel, J., Lincke, D., Vafeidis, A. T., Perrette, M., Nicholls, R. J., Tol, R. S., & Levermann, A. (2014). Coastal flood damage and adaptation costs under 21st century sea-level rise. *Proceedings of the National Academy of Sciences*, *111*(9), 3292–3297. https://doi.org/10.1073/pnas.1222469111

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hubig, N., Fengler, P., Züfle, A., Yang, R., & Günnemann, S. (2017). Detection and prediction of natural hazards using large-scale environmental data. In *International Symposium on Spatial and Temporal Databases* (pp. 300–316). Springer, Cham.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (pp. 448–456). PMLR

Ivanov, V. Y., Xu, D., Dwelle, M. C., Sargsyan, K., Wright, D. B., Katopodes, N., et al. (2021). Breaking down the computational barriers to real-time urban flood forecasting. *Geophysical Research Letters*, *48*(20), e2021GL093585. https://doi.org/10.1029/2021gl093585

Jiang, J., Liu, J., Cheng, C., Huang, J., & Xue, A. (2019). Automatic estimation of urban waterlogging depths from video images based on ubiquitous reference objects. *Remote Sensing*, *11*(5), 587. https://doi.org/10.3390/rs11050587

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall–runoff modelling using long short-term memory (LSTM) networks. *Hydrology and Earth System Sciences*, *22*(11), 6005–6022. https://doi.org/10.5194/hess-22-6005-2018

Krzysztofowicz, R., Karen, S. K., & Long, D. (1994). Reliability of flood warning systems. *Journal of Water Resources Planning and Management*, *120*(6), 906–926. https://doi.org/10.1061/(asce)0733-9496(1994)120:6(906)

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liang, Y., Jafari, N., Luo, X., Chen, Q., Cao, Y., & Li, X. (2020). WaterNet: An adaptive matching pipeline for segmenting water with volatile appearance. *Computational Visual Media*, *6*(1), 3–78. https://doi.org/10.1007/s41095-020-0156-x

Maini, R., & Aggarwal, H. (2010). A comprehensive review of image enhancement techniques. *arXiv preprint arXiv:1003.4053*.

Moy de Vitry, M., Kramer, S., Wegner, J. D., & Leitão, J. P. (2019). Scalable flood level trend monitoring with surveillance cameras using a deep convolutional neural network. *Hydrology and Earth System Sciences*, *23*(11), 4621–4634. https://doi.org/10.5194/hess-23-4621-2019

Nash, J. E., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I—A discussion of principles. *Journal of Hydrology*, *10*(3), 282–290. https://doi.org/10.1016/0022-1694(70)90255-6

NWS. (2020). Annual climate report. Retrieved from https://w2.weather.gov/climate/index.php?wfo=ilm

Pally, R., Karanjit, R., & Samadi, S. (2022). FloodDBS: Flood image DataBase system. Retrieved from https://www.kaggle.com/hhrclemson/flooding-image-dataset

Pally, R., & Samadi, S. (2022). Application of image processing and convolutional neural networks for flood image classification and semantic segmentation. *Environmental Modelling and Software*, *148*, 105285. https://doi.org/10.1016/j.envsoft.2021.105285

Paul, A., & Das, P. (2014). Flood prediction model using artificial neural network. *International Journal of Computer Applications Technology and Research*, *3*(7), 473–478. https://doi.org/10.7753/ijcatr0307.1016

Perez-Sanz, F., Navarro, P. J., & Egea-Cortines, M. (2017). Plant phenomics: An overview of image acquisition technologies and image data analysis algorithms. *GigaScience*, *6*(11), gix092. https://doi.org/10.1093/gigascience/gix092

Phillips, R. C., Samadi, S. Z., & Meadows, M. E. (2018). How extreme was the October 2015 flood in the Carolinas? An assessment of flood frequency analysis and distribution tails. *Journal of Hydrology*, *562*, 648–663. https://doi.org/10.1016/j.jhydrol.2018.05.035

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 234–241). Springer.

Samadi, S., Tufford, D., & Carbone, G. (2018). Estimating hydrologic model uncertainty in the presence of complex residual error structures. *Stochastic Environmental Research and Risk Assessment*, *32*, 1259–1281. https://doi.org/10.1007/s00477-017-1489-6

Shen, C. (2018). A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resources Research*, *54*(11), 8558–8593. https://doi.org/10.1029/2018wr022643

Sorensen, T. A. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. *Kongelige Danske Videnskabernes Selskab, Biologiske Skrifter*, *5*, 1–34.

Tabas, S. S., & Samadi, S. (2022). Variational bayesian dropout with a Gaussian prior for recurrent neural networks application in rainfall-runoff modeling. *Environmental Research Letters*, *17*(6), 065012. https://doi.org/10.1088/1748-9326/ac7247

Tonina, D., McKean, J. A., Benjankar, R. M., Wright, C. W., Goode, J. R., Chen, Q., et al. (2019). Mapping River bathymetries: Evaluating topobathymetric LiDAR survey. *Earth Surface Processes and Landforms*, *44*(2), 507–520. https://doi.org/10.1002/esp.4513

Wang, Y., Chen, Q., & Zhang, B. (1999). Image enhancement based on equal area dualistic sub-image histogram equalization method. *IEEE Transactions on Consumer Electronics*, *45*(1), 68–75. https://doi.org/10.1109/30.754419

Windheuser, L., Karanjit, R., Pally, R., Samadi, S., & Hubig, N. C. (2022). An end-to-end flood stage prediction system using deep neural networks. Retrieved from https://zenodo.org/account/settings/github/repository/HHRClemson/Flood-stAge-predIction-thRough-Deep-Neural-Networks

Zahura, F. T., Goodall, J. L., Sadler, J. M., Shen, Y., Morsy, M. M., & Behl, M. (2020). Training machine learning surrogate models from a high-fidelity physics-based model: Application for real-time street-scale flood prediction in an urban coastal community. *Water Resources Research*, *56*(10), e2019WR027038. https://doi.org/10.1029/2019wr027038