JIRBDAI Journal of Institutional Research, Big Data Analytics and Innovation

An Explainable Reinforcement Learning Approach Integrated with Rule-Based Insights for Early Urban Flood Detection

[1*]Oyerinde Daniel Diekololuwa, [2]Oluwatobi Noah Akande, [3]Ibrahim Salihu Anka

[1,3]Department of Software Engineering, Faculty of Computing, Nile University of Nigeria, Abuja.
[2]Department of Computer Science, Faculty of Computing, Nile University of Nigeria, Abuja.
*Corresponding Author: oyerinde.daniel@yahoo.com

ABSTRACT

Urban flooding remains one of the most disruptive hazards in modern cities. While machine learning surrogates allow real-time flood simulation at street level, most data-driven systems focus on spatial prediction accuracy. They do not support operational decision-making or provide clear physical explanations as to why a specific flood alert should be issued. This leaves operators relying on fixed thresholds that do not adapt to changing conditions. This study designs an integrated framework for timely flood response decision-making by coupling a Conv2D-LSTM spatiotemporal surrogate with a model-based reinforcement learning environment and a dual-layer interpretability module. The reinforcement learning agent uses an asymmetric reward function to prioritize safety-critical detection decisions, and its performance is evaluated using precision, recall, and F1-score, with rule-based insights validated through XAI-driven perturbation analysis. The Conv2D-LSTM surrogate was built using open-source terrain and rainfall data to produce depth predictions in milliseconds. The surrogate then trained the agent over 100,000 steps using an 8-sensor observation space, with SHAP and sensitivity layers incorporated for interpretability. The model was evaluated on 382 storm sequences from Houston collected between 2017 and 2022. The surrogate closely matched physics-based flood maps, achieving a Nash–Sutcliffe Efficiency of 0.9966. The reinforcement learning decision process reached an F1-score of 0.9911, missing only four flood events. XAI-derived rule-based insights aligned with real hydrological patterns. The framework demonstrates that fast surrogate modelling and adaptive decision logic can support reliable early urban flood detection, with recommended operational deployment for emergency response during flash floods.

KEYWORDS: Explainable Artificial Intelligence (XAI), Deep Reinforcement Learning (DRL), Urban Pluvial Flooding, Operational Decision Support, Asymmetric Risk Management, Spatio-Temporal Surrogate Modeling.

I. INTRODUCTION

Urban pluvial flooding is one of the most common and destructive hazards, it affects cities worldwide. This challenge is closely tied to the way cities grow and change. When cities expand because of urbanization, the natural ground meant to absorb water is covered by concrete and asphalt. This reduces infiltration and causes rainfall to generate rapid overland flow. As surface runoff increases, existing drainage systems are clogged, and as a result, frequent rainfall events can flood streets for hours. These events lead to major disruption and economic losses, with individual floods in large urban areas sometimes exceeding billions of US dollars in damages (Qin et al., 2020). Such losses are increased by the shortcomings of existing warning systems, which frequently do not provide timely or clear information that allows people to take preventive actions (Piadeh et al., 2022). The hardest hit are always areas

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

on low ground with virtually no drainage system, no means to divert the excess water, and this set of people feel the impact of the flooding, which might leave them displaced for months.

For many years the standard approach relied on full hydrodynamic models that solve the complete Saint-Venant equations for every street and pipe. Engineers still use them for designing new infrastructure because the results are accurate. Running one simulation can take anywhere from hours to several days. It's too slow for flash floods (Piadeh et al., 2022; Aderyani et al., 2025; Zhang et al., 2024). In some regions convective storms can drop 100 mm of rain in an hour and running a full simulation can take hours or even days which puts people and infrastructure at serious risk of flooding and damage. Over the last decade, machine-learning methods have offered a powerful alternative by exploiting abundant satellite and ground-sensor data to achieve much faster predictions. Early applications of Long Short-Term Memory (LSTM) networks successfully identified temporal patterns in precipitation and land-cover data to map flood-prone zones (Piadeh et al., 2022). Subsequent hybrid architectures merging Transformers with LSTMs improved both accuracy and partial interpretability through better handling of spatio-temporal relationships (Li, et al., 2024). The latest generation of sequence-to-sequence and convolutional LSTM surrogate models can now replicate complex hydrodynamic behaviour at street level in mere seconds, delivering speed improvements of several orders of magnitude (Roy et al., 2025; Liao, 2023; Ren et al., 2025).

These fast surrogate models have made real-time forecasting technically possible, but the underlying hazard itself is getting worse. Climate change is driving up both the frequency and intensity of extreme rainfall, especially in tropical and subtropical urban areas that already face convective storms (Zhili Li, 2025). At the same time, unchecked population growth and unplanned settlement expansion continue to place ever larger numbers of people in low-lying zones with minimal drainage capacity. Operational early-warning systems require clear, explainable decisions within a one-to-three-hour window at most to remain effective (He et al., 2025). Most existing research primarily focuses only on improving accuracy for flood maps or depth forecasts, they predict but omit actual decision-making that make them usable or accounts real-world cost of errors. They lack the transparent logic needed for operators to act with confidence to rely on them (Li, 2025; Aderyani et al., 2025). These highlight the main limitations in current flood forecasting systems. (i) absence of a clear framework that turns complex spatial forecasts into actionable decisions for early flood management. (ii) optimization for standard error metrics like Mean Squared Error, or comparing against other baselines, treating all errors equally. Other limitations include facts that many deep learning models operate as black boxes, offering little insight into the physical factors behind each prediction.

This study addresses these limitations by integrating an explainable policy-based reinforcement-learning agent with a fast Conv2D-LSTM surrogate for early urban flood detection, while generating rule-based insights to support transparent decision-making.

II. RELATED WORKS

Limitation found in urban flood research exists because most current models never move beyond the prediction stage. Recent studies have shown several approaches taken by researchers to successfully improve models and algorithms to suit the production of flood maps or water depth grids. Several papers demonstrate that machine-learning surrogates work well in real time. For example, Liao (2023) used a deep convolutional neural network that processes street-level flooding in just a few seconds. Roy et al. (2025) built a seq2seq LSTM model for forecasts, with inference taking less than half a second. This speed is the main advantage. Other researchers, such as Ren et al. (2025), have applied deep interpretable learning to map flood sources. They achieved errors as low as 0.03 metres with very fast computation times. Aderyani

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

et al. (2025) used clustering methods to cut processing time by over 90 percent for large city areas. These papers show that surrogates are accurate enough for city use.

Reinforcement Learning (RL) has shown it can work for flood control, but it is rarely used as a decision-support tool for early warnings. Early work by Mullapudi et al. (2020) used Deep Q-Networks to manage gates and pumps during storms. This cut down city overflows by about 40%. Other researchers have worked on different RL versions to manage costs or link systems to live sensors (Tian et al., 2023; Qin et al., 2024). However, while these models effectively mitigate water levels during an active flood, they are rarely configured to provide the predictive lead-time required for a proactive operational response before the flood stage is reached.

It has also been shown in papers that while Explainable AI (XAI) is used to interpret complex models, it does sometimes struggle to provide enough detail for critical operational decisions. Researchers have used tools like SHAP to identify what drives flood risk in different city areas (Choubin et al., 2025; Wenhao Chu et al., 2024), and some studies have combined CNN or LSTM architectures with feature attribution to highlight important terrain features (Pradhan et al., 2023). These approaches help explain why a static flood map looks a certain way, operational teams still require clear and actionable decision-logic. This research takes a more streamlined approach, as a fast surrogate model pipes its outputs to a reinforcement learning agent that helps with decision making using 8 sensors. The agent then provides transparent insights into the exact sensor that triggered such decision, all fostering urban early flood detection.

III. METHODOLOGY

This study describes the methods used to design, implement and evaluate an explainable reinforcement learning framework for early urban flood detection in Houston. The research adopts a computational research design. The methodology is divided into two main parts: the framework and the implementation of a five-phase pipeline. Each phase implements a specific component of the framework, so the theoretical design is translated into a functional software system. The approach integrates physics-based models with deep learning surrogates and adaptive agents to address slow and non-transparent warning systems for early urban flood detection. By combining computational fluid dynamics with fast neural networks, the study establishes a transparent decision-support tool for early urban flood detection. This research focuses on pluvial flooding in Houston, Texas a city that experienced extreme events such as Hurricane Harvey in 2017, which caused over $125 billion in damage (Qin, Khakzad, & Zhu, 2020). Houston was chosen because it had concurrent extreme rainfall events and is one of the largest series of extreme precipitation events in world and well documented static raster data such as elevation and land cover needed for training.

A. Computational Framework

The system is structured into several functional layers that work together to process raw geospatial data into transparent actionable auditable insights for the operator. The Data acquisition and preprocessing layer helps handles the alignment of multi-source geospatial inputs ensuring the elevation terrain, drainage, and rainfall maps are all aligned to a single coordinate system. After data alignment, the Landlab physics engine produces the true flood depths for each scenario. These ground truth maps are then used to train the surrogate model, which provides a faster alternative to the full physics simulation. The surrogate modelling layer predicts how water depth changes over time using a spatio-temporal surrogate model (Conv2D-LSTM), which serves as the main component for forecasting flood progression within a 12-hour period. The decision-making layer is where the Reinforcement Learning agent is. It observes the surrogate outputs and helps determine the alert status by evaluating the potential risk to public safety. The last layer is the Explainable layer. It is responsible for interpretability

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

using explainable AI tools to justify the actions taken by the system and provide the transparent rule-based reports. This pipeline is organized to help achieve the main goal which is early urban flood detection; the individual components' goal is to be fast as the data of one phase is piped to other phases and the speed fosters explainable early flood detection. As illustrated in Figure 1, these layers form the core structure of the pipeline.

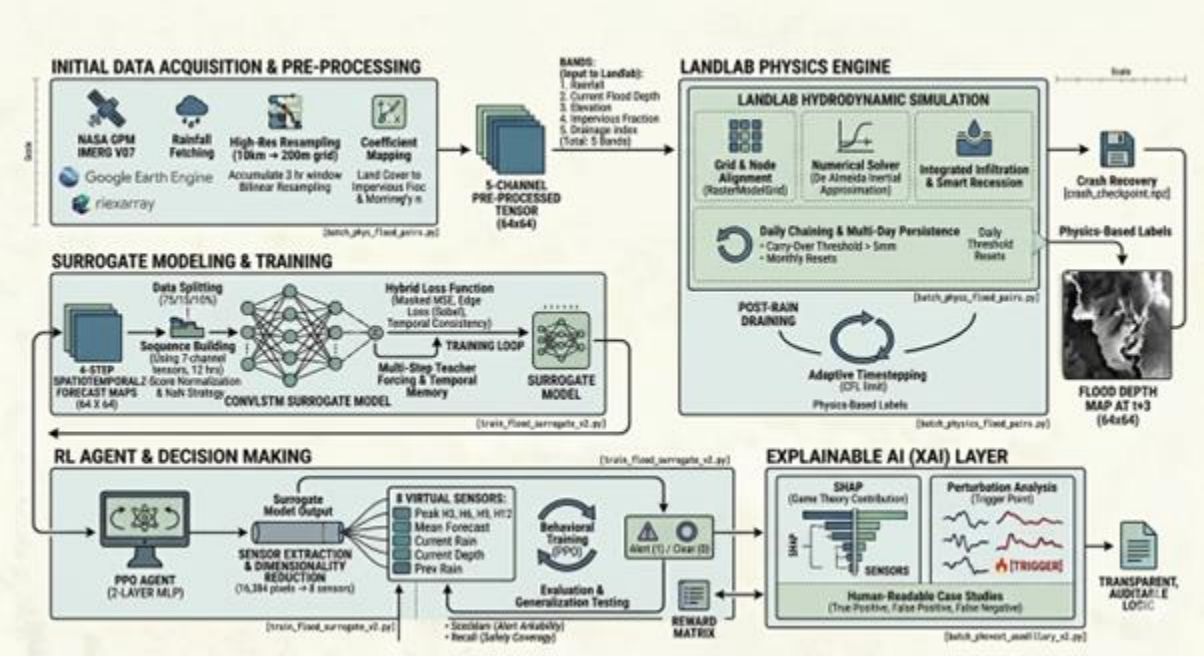

Figure 1: Computational Methodology Diagram of the Framework Generated by Gemini AI Model

The diagram in Figure 1. shows breakdown of the methodology stages of the framework. It is generated by Gemini AI model based on this work.

B. Landlab Physics-Based Ground Truth Generation

The physics engine uses the inertial approximation of the two-dimensional shallow water equations formulated by de Almeida et al. (2012).

$$\frac{\partial q_x}{\partial t} + \underbrace{\frac{\partial (q_x^2/h)}{\partial x} + \frac{\partial (q_x q_y/h)}{\partial y}}_{\text{convective acceleration}} = \text{gravity slope} - \text{friction} - \text{pressure gradients}$$

Shallow water equation is on water momentum. It has an inertial part (local acceleration) and convective acceleration. The inertial term points out that the speed of water at a point changes because conditions at that point changed (water-surface slope, depth, pressure, rainfall intensity), while the convective part describes how water momentum at a point changes when water from somewhere else flows in with a different speed. The approximation keeps it simple by omitting the convective acceleration terms but retaining the inertial terms reduces the computation load that comes with solving for velocity in both x and y directions simultaneously. Landlab Overland Flow component uses just inertia which is waterstill flows because of (i) Gravity (ii) Pressure gradients (water-surface slope) (iii) Local inertia (momentum at the point) (iv) Continuity (mass conservation). These are used to simulate water movement from cell to cell.

On a grid, the water depth in each cell changes depending on rainfall, infiltration, and how water flows between neighbouring cells. The equation is:

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

$$\frac{\partial h}{\partial t} + \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} = R(i,j) - I(i,j)$$

It is the change in water depth = rain − infiltration – net flow out of the cell.

$\frac{\partial h}{\partial t}$ is the change in water depth over time, $\frac{\partial q_x}{\partial x}$ and $\frac{\partial q_y}{\partial y}$ represent the net flow of water in the x and y directions, $R(i,j)$ is rainfall input to the cell, and $I(i,j)$ is water lost through infiltration. How water flow is computed between grid cells uses the equation below:

$$q_x^{t+1} = \frac{q_x^t - g \cdot h_f \cdot \Delta t \cdot \frac{\partial (h+z)}{\partial x}}{1 + g \cdot \Delta t \cdot n^2 \cdot |q_x^t| / h_f^{7/3}}$$

New flow = old flow + push from slope − slowing from friction, g is gravitational acceleration (9.81 m/s^2), h_f is the effective flow depth at the link between nodes, z is the bed elevation, n is Manning's roughness coefficient, and Δt is the adaptive timestep. This same equation applies in the y-direction. The effective flow depth at the link h_f which is how water cross between cells is calculated as the maximum of the water surface elevations at both neighbouring nodes minus the maximum of their bed elevations:

$$h_f = \max(h_i + z_i,\ h_j + z_j) - \max(z_i,\ z_j)$$

The surface roughness coefficient n is calculated per cell. It is using a continuous equation based on the impervious fraction f_{imp}:

$$n_{cell} = n_{pervious} + f_{imp} \cdot (n_{impervious} - n_{pervious})$$

where $n_{pervious}$ = 0.12 (vegetated areas) and $n_{impervious}$ = 0.03 (paved surfaces). A cell that is fully impervious (f_{imp} = 1.0) gets n = 0.03, it means water flows quickly across it. A cell that is fully pervious (f_{imp} = 0.0) gets n = 0.12, means vegetation slows the water down.

The adaptive timestep changes use Courant-Friedrichs-Lewy (CFL). This is what moves the simulation clock per step. It decides how big a time jump the solver is allowed to take before recalculating. If water is deep and moving fast, we use a small timestamp but if it is shallow and slow, we use a larger timestamp. This helps to prevent moving a heavy fast flow of water to quickly per step failing to account for the water in a cell because it was moving to fast.

$$\Delta t_{CFL} = \alpha \cdot \frac{\Delta x}{\sqrt{g \cdot h_{max}}}$$

Δx is the grid spacing (200 m in this implementation), h_{max} is the maximum water depth across all nodes, and alpha is a safety factor (0.2). 0.2 is the default alpha. Alpha ranges from 0.2 to 0.7. 0.2 is used because it is small enough to keep the simulation stable even on steep urban streets and also large enough that the simulation moves at a productive speed. The actual timestep used at each iteration is the minimum of Δt_{CFL} with a maximum timestep (30 seconds), and the remaining simulation time.

$$\Delta t = \min(\Delta t_{CFL},\ \Delta t_{max},\ T_{remaining})$$

So, for each 3 hours rainfall window run. Our total simulation clock is also 3 hours, at each step in the iteration, the max adaptive timestamp is clamped at 30 seconds. This means the maximum number of steps is 360 steps (10800/30). On a dry day Δt will stays at 30 seconds while during a massive flood Δt might drop to 1 or 2 seconds. It could take 5,000 to 10,000 steps to finish that same 3-hour window.

C. Adaptive Post Rain Recession

After the 3-hour rainfall phase completes we do not stop the simulation because standing water still needs to redistribute, drain, and settle into its final resting position. The same momentum equation ran at rainfall phase is used here but with the rainfall source term set to zero ($R = 0$). Also, the other active processes during the recession are infiltration (water soaking into pervious ground) and the overland flow (water moving from the high cells to the low cells under gravity). The number of recession steps used is not fixed, it is adapted based on how

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

significantly the rainfall dropped compared to the previous 3-hour window. The rain drop ratio is calculated as:

$$r_{drop} = \max\left(0, \frac{P_{t-1} - P_t}{P_{t-1}}\right)$$

P_{t-1} is the mean rainfall from the previous 3-hour window and P_t is mean rainfall from the current window. When the storm ends suddenly (e.g., 40 mm of water drops to 2 mm, $r_{drop} = 0.95$), there is a large volume of water sitting on the surface that needs many steps to drain out. When rainfall drops gradually, for example 40 mm to 35 mm, $r_{drop} = 0.125$, fewer recession steps are needed, most of the water is sustained by ongoing rain. The minimum number of recession steps is calculated using the formula below:

$$N_{min} = \max(3, 3 + \lfloor 12 \cdot r_{drop} \rfloor)$$

N_{min} gives a range from 3 steps (when rain barely decreased) up to 15 steps (when rain stopped entirely). The recession also has a smart skip condition. If rainfall is increasing from one window to the next ($P_t \geq P_{t-1}$), the recession phase is totally skipped entirely because there is no physical reason for the water to recede when more rain is arriving.

Recession runs up to a maximum of 100 steps; it will stop early once convergence is detected. The convergence is determined by comparing the maximum cell depth before and after each recession step:

$$\Delta h_{max} = h_{max}^{before} - h_{max}^{after}$$

The rule for convergence is based on the step count exceeded N_{min} and the absolute depth change falls below an adaptive threshold:

$$| \Delta h_{max} | < \tau_{conv}$$

where the convergence threshold itself adapts to the current depth:

$$\tau_{conv} = \min(\max(0.005 \cdot h_{max}, 0.001), 0.02) \text{ mm}$$

This adaptive threshold is important because a 0.01 mm change means something different depending on scale. If the maximum depth is 200 mm, a 0.01 mm change is negligible (0.005%) water has already stabilized. In a case the maximum depth is only 0.5 mm, a 0.01 mm change is significant (2%) and the water is still actively moving. Scaling the threshold with the current depth helps the recession knows to keep running longer when dealing with shallow residual water that is still finding its way to low points.

D. Temporal Carry-Over Between Windows

Each 3-hour window in the physics simulation is treated differently, each window is independent, between each window the depth map produced at the end of one window is used as the starting condition for the next. This chaining allows us to create a continuous physical timeline across an entire storm event. By carrying over the previous conditions when 30 mm of rain fell in the first 3 hours and the ground is already holding 80 mm of standing water, the same 30 mm of rain in the second window will cause much worse flooding because the water has nowhere to go. If carry-over is omitted, the model would treat every window as starting from dry ground and that would mean it would fails to capture cases where prolonged storms like Hurricane Harvey occur. Also, it helps capture recession dynamics. When rain stops, the water does not vanish. It will continue flowing from high ground to low ground over the intervals. The carry-over ensures that water in a low spot at hour 3 is still there at hour 6 even if no new rain falls and continues draining toward the bayous.

E. Simulation Output and Paired Tensor Formation

Each 3-hour window produces one depth map at the resolution of the physics grid. After the simulation of one 3-hour window at time t, the system collects a paired training sample. The input tensor $\mathbf{X}t$ is a six-band raster stack at 64 x 64 resolution:

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

$$\mathbf{X}_t = \begin{bmatrix} P_t \\ h_t \\ z \\ f_{imp} \\ D \\ P_{t-1} \end{bmatrix} \in \mathbb{R}^{6\times64\times64}$$

where P_t is the current 3-hour rainfall, h_t is the water depth at time t before the simulation runs, z is the static elevation, f_{imp} is the static impervious fraction, D is the static drainage index, and P_{t-1} is the rainfall from the previous 3-hour window. The target tensor is the single-band depth output from the physics solver:

$$\mathbf{y}_t = h_{t+3} \in \mathbb{R}^{1\times64\times64}$$

This represents the physically simulated water depth 3 hours into the future. These paired samples ($\mathbf{X}_t$, $\mathbf{y}_t$) are what the surrogate model would learn from.

a) Surrogate Modelling

The surrogate model used is the Convolutional Long Short-Term Memory network (Conv2D-LSTM) that learns to predict flood depth sequence from the paired data generated by the physics engine. It takes sequences of four consecutive input tensors ($\mathbf{X}_t$) spanning a 12-hour window and predicts the corresponding four depth maps $\mathbf{y}_t$.

b) Input Sequence Construction

The training sequences are formed by chaining four consecutive 3-hour input-target pairs that have strict temporal continuity. For a sequence starting at time t:

$$\mathbf{S} = \{(\mathbf{X}_t, \mathbf{y}_t),\ (\mathbf{X}_{t+3}, \mathbf{y}_{t+3}),\ (\mathbf{X}_{t+6}, \mathbf{y}_{t+6}),\ (\mathbf{X}_{t+9}, \mathbf{y}_{t+9})\}$$

Each pair must be separated by exactly 3 hours. If there is a gap larger than 3 hours between the consecutive pairs, it is rejected to prevent the model from learning across discontinuities. This gap enforcement is important because the ConvLSTM hidden state accumulates information between timesteps and a time gap would break that physical continuity.

c) Data Normalization

Before training, both inputs and targets are normalized using Z-score computed from a sample of the training data. For each channel c of the input:

$$\hat{X}^c_{i,j} = \frac{X^c_{i,j} - \mu_c}{\sigma_c + \epsilon}$$

μ_c and σ_c are the per-channel mean and standard deviation computed from valid (non-NaN) pixels, and epsilon = 1 x 10^{-6} prevents division by zero. The target depths are also normalized same way:

$$\hat{y}_{i,j} = \frac{y_{i,j} - \mu_y}{\sigma_y + \epsilon}$$

Areas outside the study domain are marked as NaN in the raw data. After normalization, these NaN pixels are replaced with 0.0. This is done because in a Z-scored world, 0.0 represents the "average" or "neutral" value. This means the void areas around the study domain exert no physical pull on the water during training. If they were set to the minimum value instead, the model would learn to flow water toward the borders, which is physically wrong.

F. ConvLSTM Cell Equations

To understand both Spatial and Temporal patterns, the model ConvLSTM is used. It tracks where the water is and also how the water evolves over time. ConvLSTM exposes two internal maps for every pixel on the 64 x 64 grid: (i) a long-term cell state C_t that stores the history

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

gathered over time in a 64-channel map per pixel, (ii) a short-term hidden state H_t that represents the current working knowledge. Both are 64-channel maps.

At each 3-hour window, the model uses a 3 x 3 convolution to examine each pixel and its 8 neighbours across all channels in parallel. That is a 3 x 3 over the 6 channels x 64 x 64 input grid. In total the number of maps is 64 x 64 x 64, a 64 -hidden state map per pixel multiplied by the 64 x 64 tensor from the physics simulation. The convolution controls four gating operations that update the memory:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{f}_t = \sigma\big(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{b}_f\big)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{g}_t = \tanh\big(\mathbf{W}_{xg} * \mathbf{X}_t + \mathbf{W}_{hg} * \mathbf{H}_{t-1} + \mathbf{b}_g\big)$$

σ is the sigmoid activation function, * is convolution (using 3 x 3 kernels with same padding), and W and b are the learnable weight matrices and biases used. The four gates determine what happens to the information at each step:

- i_t (Input Gate): Should this new rainfall information be saved into memory?
- f_t (Forget Gate): Should the previous memory be kept or erased?
- g_t (Candidate): What is the proposed new content based on the current input?
- o_t (Output Gate): What portion of the updated memory should be reported to the next stage?

The cell state and hidden state updates are:

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathbf{H}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

$\odot$ is element-wise multiplication. The cell state C_t is a long-term memory, it is carried across the windows, since it is a 12-hour window and the ConvLSTM looks over all steps and into the next window. When a storm persists, the forget gate is close, set to 1.0 so the model preserves the accumulated water state and when rain stops and the water level drops, the forget gate drops toward 0.0 so the model can release the old state. The hidden state H_t is the filtered version of that memory that gets passed forward to the next processing stage. At the first window (t = 3hr), both C and H are zeros. Since it is the first state, the gates set these maps with an initial value which is based on the incoming rainfall data. The window that follows (6hr, 9hr, 12hr), the gates existing maps using the new input combined with what was remembered from the previous step.

a) Stacked ConvLSTM Layers

The concept of multi-layer in each window is organised around the hidden state maps, with the three ConvLSTM layers, each with 64 hidden channels per pixel. Each layer has its own independent set of gates and its own memory maps:

- Layer 1 (input: 6 channels, output: 64 channels): Processes the raw input bands and extracts basic spatial features like rainfall intensity gradients and terrain edges.
- Layer 2 (input: 64 channels, output: 64 channels): Takes Layer 1's output and learns the higher-order patterns like flow accumulation and ponding behaviour.
- Layer 3 (input: 64 channels, output: 64 channels): Produces the final representation that captures the full spatial-temporal state of the flood.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

Each layer has its own gates and processes the input in sequence, the first layer handling the raw input and forwarding its hidden state to Layer 2, this then passes it to Layer 3. This repeated across all 3-hour windows for over the 12-hour sequence (hour 3,6,9,12). The model returns the hidden state from Layer 3 at every timestep, so the loss function can then evaluate all four predicted frames rather than just the final one. This forces the model to maintain an accurate internal state at every intermediate step rather than only getting the final answer right.

b) Convolutional Decoder

The 64-channel hidden state produced by Layer 3 is an internal representation that the model uses for its own computations. The goal is to convert this into a usable flood depth prediction, so the convolutional decoder converts the 64 channels down to a single depth value per pixel:

$$\hat{\mathbf{y}}_t = \text{Conv}_{1\times1}\Big(\text{LReLU}\Big(\text{BN}\Big(\text{Conv}_{3\times3}\Big(\text{LReLU}\big(\text{BN}(\text{Conv}_{3\times3}(\mathbf{H}_t))\big)\Big)\Big)\Big)\Big)$$

The decoder reduces the channels from 64 to 32 to 16 to 1 through three convolutional layers. Each intermediate layer uses batch normalization and Leaky ReLU activation (slope 0.1) to keep gradients flowing through negative values. The final layer outputs a single channel with no activation function because the targets are Z-normalized and can take negative values.

c) Loss Function and Training

At the end of each 12-hour window, a loss is calculated using the $\mathcal{L}_{total}$ in the equation below, this is used to update the gates parameters in each layer and improves it over time. The $\mathcal{L}_{total}$ formula is stated below:

$$\mathcal{L}_{total} = 1.0 \cdot \mathcal{L}_{MSE} + 0.1 \cdot \mathcal{L}_{edge} + 0.05 \cdot \mathcal{L}_{temp}$$

$\mathcal{L}_{MSE}$ stands for mean squared error that helps decide the pixel-wise prediction accuracy only on valid land pixels, it ignores the invalid border:

$$\mathcal{L}_{MSE} = \frac{\sum_{i,j} M_{i,j} \cdot (\hat{y}_{i,j} - y_{i,j})^2}{\sum_{i,j} M_{i,j}}$$

$\mathcal{L}_{edge}$ is an edge-aware loss that preserves sharp flood boundaries and avoids smooth average predictions that minimise MSE but erase the critical spatial transitions where water rises from 0 mm to 200 mm over a single cell. It uses Sobel filters to compare gradients of the predicted and true depth maps:

$$\mathcal{L}_{edge} = \frac{\sum_{i,j} M_{i,j} \cdot \left[(\hat{G}_x - G_x)^2 + (\hat{G}_y - G_y)^2\right]}{\sum_{i,j} M_{i,j}}$$

$\mathcal{L}_{temp}$ is a temporal consistency loss that penalises unphysical jumps between consecutive frames. If the true depth changes by 20 mm between two intervals but the model predicts a 170 mm jump, this loss catches the such errors:

$$\mathcal{L}_{temp} = \frac{\sum_{t=2}^{T} \sum_{i,j} M_{i,j}^t \cdot \left[(\Delta\hat{y}_{i,j}^t - \Delta y_{i,j}^t)^2\right]}{\sum_{t=2}^{T} \sum_{i,j} M_{i,j}^t}$$

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

Weights (1.0, 0.1, 0.05) prioritize accurate depths first, sharp boundaries second, smooth temporal evolution third. The training parmaters are (i) AdamW optimizer with learning rate of 5 x 10^{-4} (ii) weight decay is 1 x 10^{-4} (iii) gradient clipping at norm 1.0. The learning rate is halved when validation loss plateaus for 8 epochs also data is splatted into training (75%), validation (15%), and testing (10%) and augmentation are done via random flips and 90-degree rotations applied across all timesteps in a sequence. Early stopping halts training after 20 epochs without improvement, and the checkpoint with the lowest validation loss is saved for deployment in the RL environment.

G. Reinforcement Learning

The alerting agent is formulated as a Markov Decision Process (MDP). The trained surrogate is the environment dynamics. The MDP is defined by the tuple (S, A, T, R, gamma).

a) State Space Definition

The state space S consists of an eight-dimensional observation vector derived from the surrogate model output and the current input conditions. For a given storm scenario at time t, the surrogate first predicts four depth maps corresponding to +3, +6, +9, and +12 hours. These four 64 x 64 maps (totalling over 16,000 values) are then compressed into 8 scalar features:

$$\mathbf{s}_t = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \end{bmatrix} = \begin{bmatrix} \max(\hat{y}_{t+3}) \,/\, \tau \\ \max(\hat{y}_{t+6}) \,/\, \tau \\ \max(\hat{y}_{t+9}) \,/\, \tau \\ \max(\hat{y}_{t+12}) \,/\, \tau \\ \text{mean}(\hat{y}_{t+3:t+12}) \,/\, \tau \\ \bar{P}_t^z \\ \bar{h}_t^z \\ \bar{P}_{t-1}^z \end{bmatrix}$$

where τ = 150 mm is the safety threshold used as a normalizing constant for the peak-based sensors (s_1 through s_5), and the superscript z is the Z-score normalized real-time values. Sensors s_1 to s_4 report the deepest predicted pixel at each forecast window. Sensor s_5 provides the mean forecast across the 12-hour window. Sensors s_6, s_6, and s_8 report current rainfall, current depth, and previous rainfall.

b) Action Space Definition

The action space A is discrete with two actions:

$$A = \{0,1\}$$

a = 0 means "remain quiet" and a = 1 means "issue a public flood alert." Each episode is a single-decision event: the agent observes the 8-dimensional state, takes one action, receives a reward, and the episode terminates.

c) Reward Function

The reward function R(s, a) is the asymmetric cost structure of emergency management. The ground truth label is determined by whether the maximum depth across all four target maps exceeds the threshold:

$$\text{TrueFlood} = \begin{cases} \text{True} & \text{if } \max(y_{\text{true}}) > \tau \\ \text{False} & \text{otherwise} \end{cases}$$

The reward is then:

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

$$R(a,\text{TrueFlood}) = \begin{cases} +50.0 & \text{if } a = 1 \text{ and TrueFlood} = \text{True} \quad \text{(True Positive: Correct Alert)} \\ -250.0 & \text{if } a = 0 \text{ and TrueFlood} = \text{True} \quad \text{(False Negative: Missed Flood)} \\ -100.0 & \text{if } a = 1 \text{ and TrueFlood} = \text{False} \quad \text{(False Positive: False Alarm)} \\ +20.0 & \text{if } a = 0 \text{ and TrueFlood} = \text{False} \quad \text{(True Negative: Correct Quiet)} \end{cases}$$

The penalty for a missed flood (−250) is 2.5 times larger than for a false alarm (−100), which helps capture the idea that under-warning is more costly than over-warning. The lower reward for correct quiet (+20 vs +50) prevents the agent from becoming complacent and defaulting to quiet on unclear scenarios.

H. Balanced Training Dataset

To prevent a case of always-alert or always-quiet policy, episodes are sampled with equal probability from flood and non-flood scenarios.

$$\text{flood\_indices} = \{i: \max(y_i) > \tau\}$$

$$\text{quiet\_indices} = \{i: \max(y_i) \leq \tau\}$$

At each episode, the sampling probability is:

$$P(\text{select flood}) = P(\text{select quiet}) = 0.5$$

d) Policy and Value Networks

The PPO agent uses a Multi-Layer Perceptron (MLP) policy with two hidden layers of 128 neurons each:

$$\pi_\theta(s) = \text{softmax}(W_3 \cdot \text{ReLU}(W_2 \cdot \text{ReLU}(W_1 \cdot s + b_1) + b_2) + b_3)$$

where $W_1 \in \mathbb{R}^{128\times 8}$, $W_2 \in \mathbb{R}^{128\times 128}$, and $W_3 \in \mathbb{R}^{2\times 128}$. The value network $V_\phi(s)$ shares the same architecture but outputs a single scalar with $W_3' \in \mathbb{R}^{1\times 128}$. The two networks share no weight.

e) PPO Objective Function

The agent is trained using the PPO clip objective from Schulman et al. (2017):

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta) \cdot \hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \cdot \hat{A}_t\right)\right]$$

where $r_t(\theta)$ is the probability ratio between the new and old policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$$

and a_t is the Generalized Advantage Estimate (GAE):

$$\hat{A}_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l}$$

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

with gamma = 0.99, lambda = 0.95, and epsilon = 0.2. The clipping parameter constrains the policy ratio to [0.8, 1.2], preventing the agent from over-reacting to a single experience. The full PPO loss also includes a value function loss and an entropy bonus:

$$\mathcal{L}_{PPO} = -L^{CLIP}(\theta) + c_1 \cdot \mathcal{L}_{value} - c_2 \cdot H[\pi_\theta]$$

where $c_1 = 0.5$ and $c_2 = 0.01$. The entropy bonus prevents premature convergence to a deterministic policy. For a binary action space, a small entropy coefficient is sufficient to prevent the agent from committing fully to one action in early training.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

PPO Training Hyperparameters
The following training parameters were used:

a) Total training steps: 100,000
b) Rollout buffer size (n_{steps}): 512
c) Mini-batch size: 64
d) Number of epochs per update (n_{epochs}): 10
e) Learning rate: 3 x 10^{-4}
f) Discount factor (gamma): 0.99
g) GAE lambda: 0.95
h) Clip range (epsilon): 0.2
i) Entropy coefficient: 0.01

I. Explainability Layer: SHAP and Local Sensitivity Analysis

The explainability component provides two forms of evidence for each agent decision. Every alert or quiet outcome can be traced back to specific sensors and their contributions.

a) SHAP Attribution

SHAP values are based on cooperative game theory from Shapley (1953). The Shapley value for sensor i measures the average marginal contribution of that sensor across all possible sensor sets:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\ (|N|-|S|-1)!}{|N|!} [v(S \cup \{i\}) - v(S)]$$

N is the set of all 8 sensors, S is a subset excluding sensor i, and $v(S)$ is the agent's output when only sensors in S are active. A positive SHAP value means the sensor pushed toward the alert decision; a negative value means it pushed against it. The KernelExplainer approximates these values using 100 sampled sets against a background dataset of 50 observations.

b) Local Sensitivity Analysis (Decision Trigger Detection)

SHAP identifies which sensors contributed the most evidence, The local sensitivity analysis helps identify which sensor is the actual tipping point that would flip the decision. For each sensor i, the analysis perturbs its value to 10% and 200% of its current reading while keeping all others fixed:

$$a_{low} = \pi_\theta(s^{(i \leftarrow 0.1 \cdot s_i)}), a_{high} = \pi_\theta(s^{(i \leftarrow 2.0 \cdot s_i)})$$

$s^{(i \leftarrow 2.0 \cdot s_i)}$ means the observation vector with sensor i replaced by value v while all other sensors are kept constant. The local sensitivity score for sensor i is computed as:

$$\text{LSA}_i = \text{flip\_score}_i + \text{weight}_i$$

where:

$$\text{flip\_score}_i = \begin{cases} 2 & \text{if } a_{low} \neq a_{base} \\ 0 & \text{otherwise} \end{cases} + \begin{cases} 2 & \text{if } a_{high} \neq a_{base} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{weight}_i = \frac{|s_i|}{\max_j |s_j| + \epsilon}$$

A score of 2.0 or above marks the sensor as a TRIGGER noting that it is a sensor that alone is enough to flip the decision.

J. Rule-Based Statement Generation

The rule based uses a formal mapping function f_{XAI}. This function translates the SHAP and LSA scores into statements. The process is defined as a mapping from the decision outcome a, the ground truth Y_{true}, and the primary feature attributions ϕ_{max} and LSA_{max}.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

The Narrative Statement S is generated as:

$$S = f_{XAI}(a, Y_{true}, sensor_{SHAP}, sensor_{LSA})$$

$sensor_{SHAP} = \text{argmax}_i \mid \phi_i \mid$ represents the primary evidence, and $sensor_{LSA} = \text{argmax}_i(LSA_i)$ represents the primary decision trigger. The mapping logic follows a formal Decision Case Matrix shown in Table 1:

Table 1: XAI Narrative Generation Decision Matrix

Case	Logical Condition	Narrative Rule (Reason / Trigger)
True Positive	$a = 1, \text{Max}(Y_{true}) > \tau$	Evidence provided by ϕ_{max}; Triggered by LSA_{max}.
False Positive	$a = 1, \text{Max}(Y_{true}) \leq \tau$	Over-weighting of ϕ_{max}; Instability detected in LSA_{max}.
False Negative	$a = 0, \text{Max}(Y_{true}) > \tau$	Insufficient evidence from ϕ_{max}; Awaiting commitment from LSA_{max}.
True Negative	$a = 0, \text{Max}(Y_{true}) \leq \tau$	No risk evidence in ϕ_{max}; LSA_{max} stable within safe limits.

Narrative Rule Generation

a) **Input:** Agent Action a, Ground Truth Y_{true}, Sensor SHAP values ϕ, Sensor LSA scores LSA_i.

b) **Identification Step:**
   i. Primary Evidence: $sensor_{SHAP} = \text{argmax}_i \mid \phi_i$
   ii. Primary Trigger: $sensor_{LSA} = \text{argmax}_i(LSA_i)$

c) **Case Mapping Step:**
   i. **Case 1 (True Positive):** If $a = 1$ and $\text{Max}(Y_{true}) > \tau$, then Rule = "Evidence in $sensor_{SHAP}$; Triggered by $sensor_{LSA}$."
   ii. **Case 2 (False Positive):** If $a = 1$ and $\text{Max}(Y_{true}) \leq \tau$, then Rule = "Over-weighting of $sensor_{SHAP}$; Instability in $sensor_{LSA}$."
   iii. **Case 3 (False Negative):** If $a = 0$ and $\text{Max}(Y_{true}) > \tau$, then Rule = "Insufficient evidence from $sensor_{SHAP}$; Awaiting $sensor_{LSA}$."
   iv. **Case 4 (True Negative):** If $a = 0$ and $\text{Max}(Y_{true}) \leq \tau$, then Rule = "No risk in $sensor_{SHAP}$; $sensor_{LSA}$ stable."

d) **Output:** Narrative Statement S.

Every sentence is mapped to the sensor that provided the most mathematical evidence (SHAP) and the sensor that acted as the final tipping point (LSA). To validate that these explanations are faithful to the agent's policy, we implement a Top-K Fidelity metric. For a given observation s, the primary sensor $k = \text{argmax}_i \mid \phi_i \mid$ is set to zero to produce $s_{ablated}$ where $s_k = 0$. The fidelity score F is defined as the absolute change in the policy's alert probability:

$$F = \mid \pi_\theta(a = 1 \mid s) - \pi_\theta(a = 1 \mid s_{ablated}) \mid$$

High mean F across shows that SHAP correctly identifies the decision-critical features. Stability of these rankings is also measured using the Spearman Rank Correlation (ρ) across multiple independent runs to ensure the attributions remain consistent across different background references.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

K. Integration of the Framework

The framework is divided into a five-phase pipeline. Python is the language used to train the models used. At the preprocessing stage, the raster is merged and reprojected using GDAL tools from OSGeo4W. QGIS helped verify that the raster is aligned correctly.

a) Data Collection and Pre-processing

4 static rasters used to cover Houston area, are listed in Table 2. Each raster was aligned and reprojected to UTM Zone 15N and resampled to a 200 m grid.

Table 2: Geospatial Data Layer Specifications and Pre-processing

S/N	Dataset	Source Agency	Original Resolution	Target Resolution	Resampling/Processing Technique
1.	Digital Elevation Model (DEM)	USGS (Earth Explorer)	30 m	200 m	Bilinear Interpolation
2.	Precipitation (Rainfall)	NASA (GPM-IMERG)	0.1° (~11 km)	200 m	Bilinear Interpolation & 3hr Summation
3.	Drainage Network	Derived (OpenStreetMap)	200 m	200 m	Rasterization & Binary Masking
4.	Impervious / Manning's *n*	ESA (WorldCover)	10 m	200 m	Mode Resampling

b) Data Processing Pipeline

Resulting raster layers are aligned to a common 200 m grid covering the Houston study area (64 × 64 cells, approximately 12.8 km × 12.8 km):

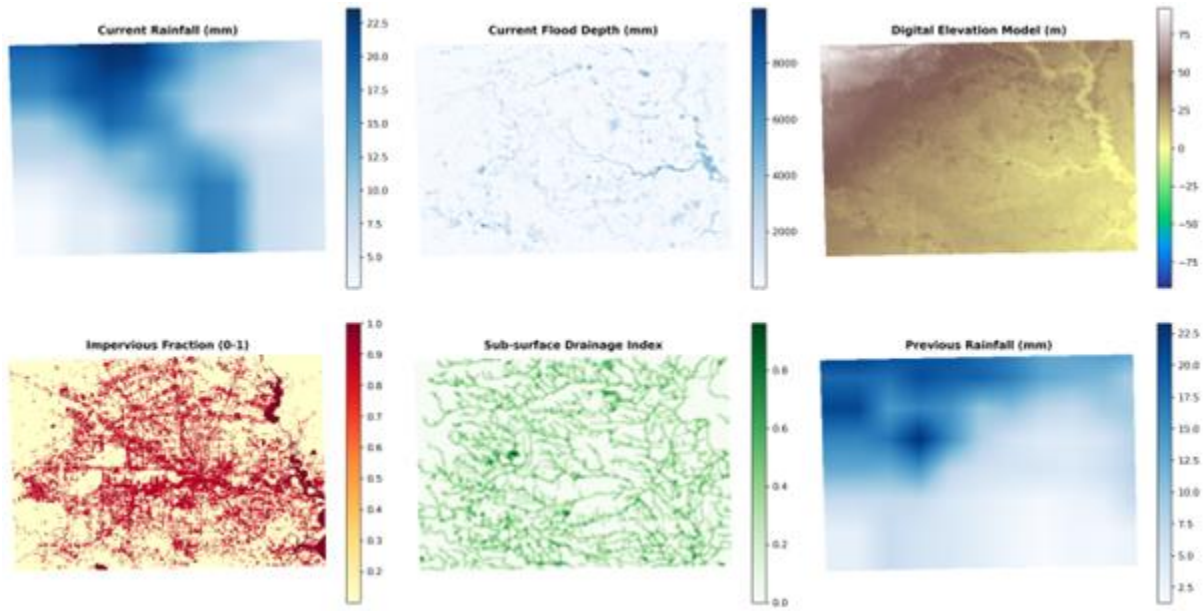

Figure 2: Visualization of the 6-channel input state sampled during the peak of Hurricane Harvey (August 28, 2017).

The resulting 6-channel tensor serves as the high-fidelity input for the surrogate model training. The use of six distinct physical channels allows the model to learn complex relationships

https://doi.org/10.5281/zenodo.20388429

between static features (topography and land use) and dynamic temporal features (current and previous rainfall pulses), leading to more accurate spatial-temporal flood propagation predictions.

Example: Reproject and resample DEM to 200 m
gdalwarp -tr 200 200 -r bilinear -t_srs EPSG:32615 input_dem.tif aligned_dem.tif
Land-cover classes from ESA WorldCover are first mapped to impervious fractions using a lookup table, and then reclassified into Manning's n values (e.g., 0.03 for paved surfaces, 0.09 for grasslands, and 0.12 for dense vegetation) using Python (rasterio and numpy). The impervious fraction is used to determine runoff potential by assigning higher values to built-up areas. Rainfall rasters from GPM-IMERG are summed into 3-hour windows and interpolated using the bilinear method. Drainage data from OpenStreetMap is rasterized to represent drainage capacity per cell. Normalization is then applied using the mean and standard deviation from the training set to scale the sensor inputs for model training.

a) Physics-Based Simulation and Ground Truth Generation (Phase 1)
This stage creates the accurate "ground truth" data that the surrogate model will learn from. A physics-based simulation is necessary because it produces realistic flood behaviour based on actual physical laws of water movement. Landlab OverlandFlow module is used for physics simulation, it uses the inertial approximation of the Shallow Water Equations by de Almeida et al. (2012). This phase is vital because without the reliable reference data, the surrogate would learn incorrect patterns and give wrong predictions. The goal here is to generate high-quality, multi-step flood depth (target depth maps) sequences that capture how water spreads over time in a real urban environment. These sequences become the target outputs that the Conv2D-LSTM surrogate is trained to predict quickly.
The pre-processed data are aligned on a common grid at a spatial resolution of 200 metres per cell. This size allows the model to include major urban flow paths and drainage features in each cell. At the same time, the total number of cells stays low enough that the physics simulations and later RL training can run in a reasonable amount of time across thousands of episodes. Land-cover classes from the ESA WorldCover dataset are mapped to Manning's n values ranging from 0.03 for smooth urban surfaces (asphalt, concrete) to 0.12 for vegetated areas. The mapping is used because Manning's n determines how much the terrain slows water movement. Smoother surfaces let water flow faster. Rougher or vegetated areas slow it down a lot. Impervious fractions come from the percentage of built-up pixels in each cell. That percentage tells how much rain becomes surface runoff instead of soaking in. Built-up areas send almost all the rain straight into runoff. Vegetated areas let some of it infiltrate.
Rainfall gets added up into 3-hour windows because rain it is a dynamic inputs. It needs to be recorded at time interval to show how wet the soil is before the storm hits. The actual water movement is calculated using Landlab's OverlandFlow tool. It follows standard physical rules and uses small time steps to keep the results stable. Manning's n is applied per cell from the reclassified land-cover layer. Infiltration is set at 5 mm/hour on pervious cells using a constant rate, with full runoff on impervious cells. Open boundaries allow free outflow at the edges of the domain to simulate water leaving the study area. Rainfall is applied as a distributed input across the grid from the GPM-IMERG windows. Simulations run for 12 hours per event, recording depths at 3-hour intervals (t=0, 3, 6, 9, 12 hours) to create multi-step ground-truth sequences. The 12-hour duration covers the full lifecycle of typical convective storms in Houston, while 3-hour intervals provide enough temporal resolution to capture rapid onset and recession. Each sequence is saved as a tensor ($64 \times 64 \times 4$) for training. The output depth maps are normalized using pre-computed statistics. This process generates 1,111 event sequences used for surrogate training.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

b) Spatio-Temporal Surrogate Model Implementation (Phase 2)

The purpose of this stage is to train a fast neural network that can imitate the slow physics-based model. The ground truth data generated in Phase 1 helps the system learn how to mimic realistic flood behaviour without the high computational cost of the physical equations**.** The surrogate is needed because the full physics model is too slow to be used thousands of times during RL training. A fast surrogate allows the agent to practice many episodes in a reasonable time by predicting flood maps in milliseconds rather than minutes.

The Conv2D-LSTM architecture takes six-channel input tensors (current rainfall, existing depth, terrain elevation, impervious fraction, drainage capacity, prior rainfall accumulation) at 64×64 resolution for four timesteps. The model consists of three ConvLSTM layers (64 filters, 3×3 kernels, padding='same') followed by a three-layer convolutional decoder block that reconstructs the predicted depth maps for the next four timesteps (64 × 64 × 4).

The composite loss function includes:

i. Mean Squared Error (MSE) for pixel-wise accuracy
ii. Edge-aware term (Sobel gradient difference) for boundary preservation
iii. Temporal consistency term (Squared Error difference between consecutive predicted frames) for smooth evolution AdamW optimizer is used with learning rate 0.0005, weight decay $1e^{-4}$, and ReduceLROnPlateau scheduler (factor 0.5, patience 8 epochs). Training runs for up to 200 epochs with early stopping on validation loss. Batch size is 8. The trained model generates predictions in milliseconds, making it feasible to run thousands of episodes during RL training.

c) Reinforcement Learning Environment and Agent Training (Phase 3)

This stage trains the intelligent decision-making agent. The RL environment is built around the surrogate so the agent can practice a hundred thousand times without waiting for slow physics simulations. This is the core of the autonomous system. The diagram shows the interaction between the trained surrogate and the decision-making agent. Conv2d-LSTM provides the state needed by RL agent and interacts with the environment, at each step, the agent receives an observation vector state containing forecast peak depths (3, 6, 9, 12 hours), average forecast depth, current rainfall intensity, current surface depth, and antecedent rainfall accumulation. The binary action space is: 0 (no alert) or 1 (alert) with the reward applied after each prediction based on if the maximum predicted depth exceeds 150 mm. The rewards used are +50 for correct alert, +20 for correct non-alert, –100 for false alarm, –250 for missed flood.

d) Explainability and Rule Generation (Phase 4 & 5)

The final stage of the framework is the Explainability and Rule-Based layer. The Rule based generation is achieved using both SHAP and Local Sensitivity Analysis. SHAP ranks sensors based on its influence, the higher it influences the higher it is ranks on the graph while the Local sensitivity analysis is a sensitivity score of 2.0 or higher used to mark the critical tipping points. The result from the dual sensitivity analysis is used to build clear statements for rule-based generation. The system identifies the top contributors from both analyses to explain why a flood was detected or why a false alarm occurred. For a successful warning, an example of a rule-based statement "Reason: This alert is driven by the 3-hour peak forecast. Trigger: The decision is confirmed because the 12-hour peak forecast is identified as the final tipping point for the alert." For a false positive, it says: "Reason: The false alarm happened because the system reacted too strongly to the current water depth sensors. Insights: The decision was sensitive to a minor change in rainfall, which became a false trigger for the alert." These explanations come from mapping the top SHAP and Local sensitivity sensors helping generate the human readable statement used.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

L. Performance Evaluation of the Framework

For evaluation, two separate datasets are used to ensure that the results are not influenced. The primary dataset consists of 1,111 storm sequences (4,444 total rasters / 4 windows) used during the training phase. For the surrogate model, this data was split into 80% for training, 10% for validation, and 10% for internal testing. The reinforcement learning agent was trained using the full 1,111 sequences, the surrogate provided 1,111 storm sequence state that the RL agent used for its training. To conduct a strict evaluation of the system, a completely separate external dataset of 382 sequences (1,528 total rasters / 4 windows) from 2021–2022 was used. These 382 sequences were kept entirely out of the training process for both the surrogate and the RL agent. The following are the metrics used:

(a). Nash-Sutcliffe Efficiency (NSE) compares predictions to the observed mean as a benchmark. It used because it penalises big errors more

$$NSE = 1 - \frac{\sum(observed - predicted)^2}{\sum(observed - mean_{observed})^2}$$

(b). Root Mean Square Error (RMSE) gives the average error magnitude in millimetres for depth predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum(observed - predicted)^2}$$

(c). Precision measures the fraction of issued alerts that are correct.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

(d). Recall shows what fraction of actual floods the system detects.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

(e). The F1-score provides a balanced average of both precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

IV. RESULTS AND DISCUSSION

The final tests on 382 new storm sequences gave clear results for every part of the framework.

A. Surrogate Model Performance and Spatial Validation

The surrogate model performance is summarized in the following table.

Table 3: *Surrogate Model Performance Summary*

Metric	Value	Interpretation
Nash-Sutcliffe Efficiency	0.9966	Excellent (threshold > 0.9)
Coefficient of Determination	0.9966	99.66% variance explained
Mean Absolute Error (MAE)	1.63 mm	Average prediction error
Root Mean Square Error	2.33 mm	Well below 20 mm target

Table 3. shows that the surrogate model matches the Landlab physics results. The Nash Sutcliffe Efficiency of 0.9966 exceeds the 0.90 benchmark for strong performance, while the R^2 of 0.9966 indicates that nearly all the variations in water depth are captured. Errors remain minimal, with a Mean Absolute Error of 1.63 mm and a Root Mean Square Error of 2.33 mm, both well below the 20 mm threshold for urban street flooding. These results confirm that the surrogate can reliably reproduce flood dynamics and provide a fast environment for training the reinforcement-learning agent.

https://doi.org/10.5281/zenodo.20388429

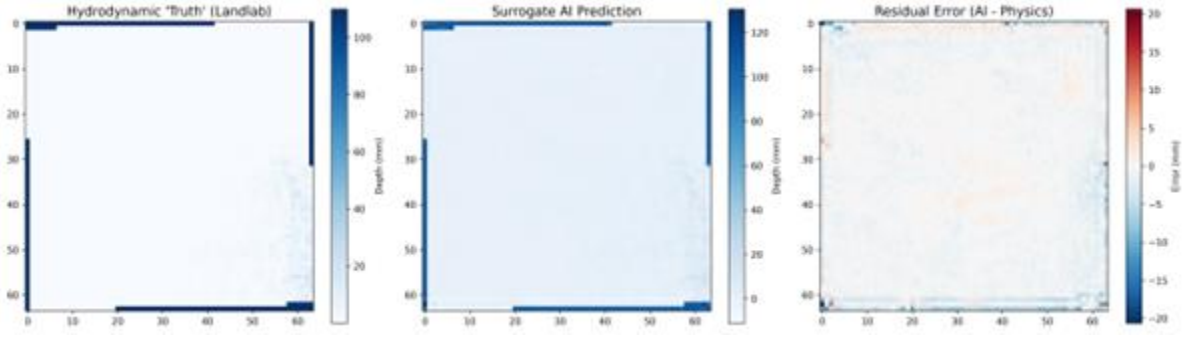

Figure 3: Spatial Comparison – Low Flood Scenario (Sequence #0) (Ground truth left, prediction right, low inundation.)

This is a low-intensity scenario from the test dataset. It shows close visual match in spatial distribution for a case in the test data where the ground truth as low inundation. The surrogate correctly identified limited ponding areas. The hydrodynamic Truth map (left) is the ground truth for a low-intensity flood case from the test set. The second map shows the depth predicted by the surrogate model, while the map on the right is the error map ranging from +20 to -20 (mm), displaying the spatial difference between the Ground Truth and Predicted values.

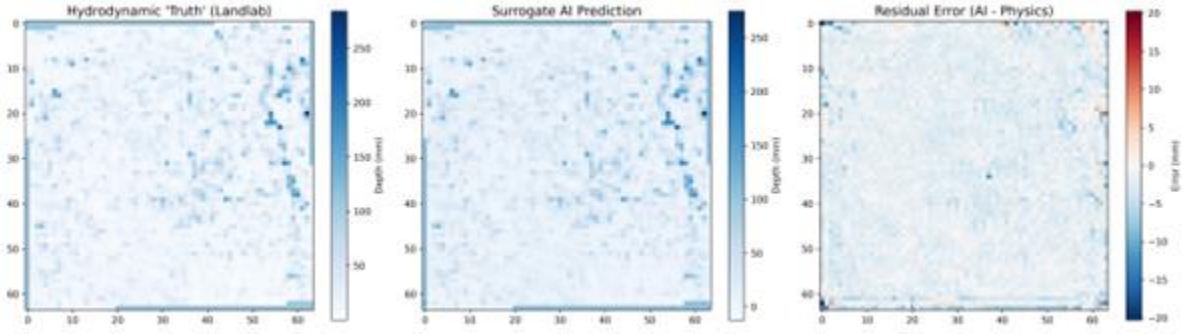

Figure 4: Spatial Comparison – Moderate Flood Scenario (Sequence #191) (Ground truth left, prediction right, moderate inundation.)

The moderate-intensity scenario illustrates case in the test data where the ground truth as case in the test data where the ground truth as low inundation. The surrogates can reproduce more complex flood patterns and extended ponding areas. The hydrodynamic Truth map (left) is the ground truth for a standard inundation case with moderate water depths. The second map displays the depth predicted by the surrogate model, while the map on the right is the error map ranging from +20 to -20 mm, showing the spatial difference between the Ground Truth and Predicted values.

B. Reinforcement Learning Agent Training and Predictive Accuracy

The RL agent performance is summarized in Table 4:

Table 4: RL Agent Performance Summary

Metric	Value	Interpretation
Total Test Episodes	382	Unseen 2021-2022 data
Average Episode Reward	+42.77	Positive indicates success
Precision (Alert Class)	0.9940	99.4% of alerts were correct
Recall (Alert Class)	0.9882	98.8% of floods were caught

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

F1-Score	0.9911	99.1% balanced performance
Overall Accuracy	0.9843	98.4% correct decisions

Table 3. shows the evaluation performance of the PPO-based reinforcement learning agent in binary flood alerting (issue alert or remain silent) on the 382 unseen test sequences from 2021–2022 storm events in Houston. These metrics are standard classification indicators adapted for alerting tasks, where the positive class is "alert" (predicted flood risk > 150 mm).
Total Test Episodes (382) refers to the number of independent flood/non-flood sequences used for final evaluation, ensuring assessment on data not seen during training. Average Episode Reward (+42.77) is the mean total reward per episode with the asymmetric scoring. Correct alerts get +50, correct non-alerts get +20, false alarms cost -100, and missed floods cost -250. This score shows the policy works well overall. Precision (0.9940) tells what fraction of issued alerts were correct. True Positives divided by True Positives plus False Positives. It means the alerts are highly reliable and avoid unnecessary warnings. Recall (0.9882) shows what fraction of actual floods were caught. True Positives divided by True Positives plus False Negatives. It shows strong detection of real risk events. F1-Score (0.9911) is the balanced average of precision and recall. It gives a solid measure of overall alerting performance.

Table 5: Confusion Matrix Results

	Predicted No Alert	**Predicted Alert**
Actual No Flood (44)	42	2
Actual Flood (338)	4	334

Table 5. summarizes the performance of the PPO-based reinforcement learning agent in binary flood alerting (issue alert or remain silent) across the 382 unseen test sequences from 2021–2022 Houston storm events.

(a). True Positives (334) indicate cases where the agent correctly issued an alert when a flood occurred.
(b). True Negatives (42) indicate cases where the agent correctly remained silent when no flood occurred.
(c). False Positives (2) are cases where the agent issued an alert despite no flood occurring.
(d). False Negatives (4) are cases where the agent failed to issue an alert when a flood occurred.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

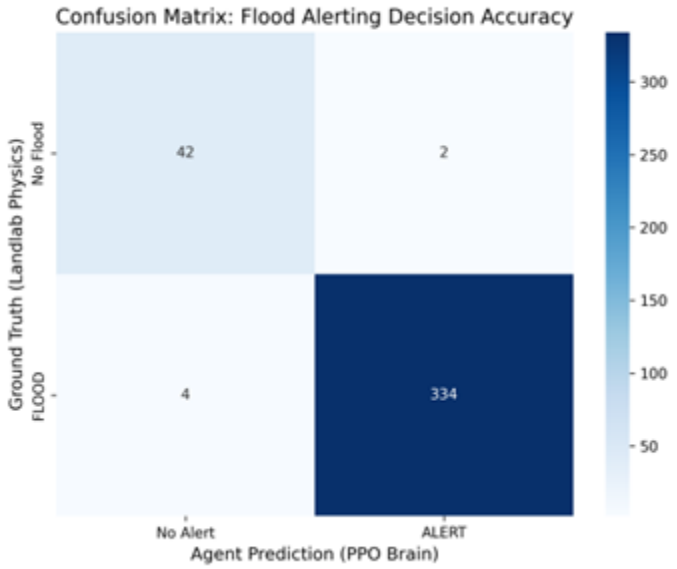

Figure 5: Confusion Matrix Heatmap.

The heatmap highlights the confusion matrix results in Table 4.

C. Rule-Based Insights

The dual XAI layer produces interpretable audits for all outcome types (True Positive, True Negative, False Positive, False Negative). SHAP values show how much each sensor influences the final decision. Local sensitivity analysis finds the critical triggers, places where a small change in input could flip the outcome the other way forming our rule-based explanations. The figures below are picked from certain scenarios from the 382 unseen test cases.

(a). True Positive Case Scenario Study

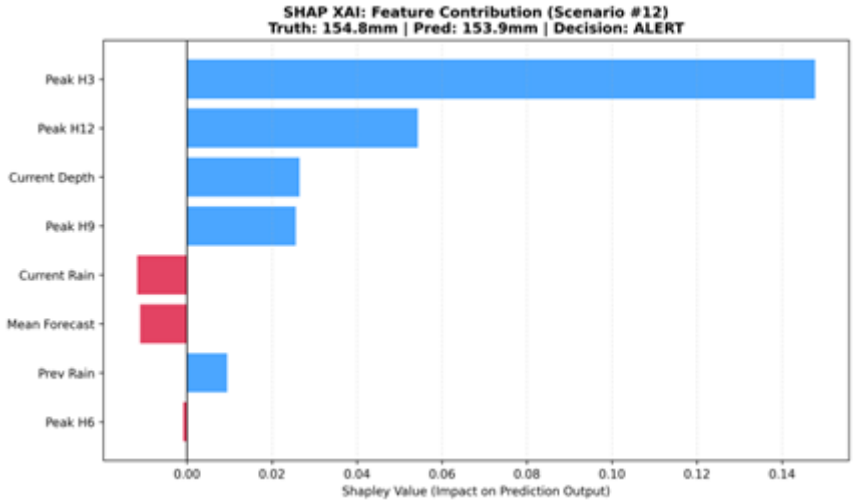

Figure 6: SHAP Feature Contribution - True Positive Case Study (Scenario #12)

The chart shows SHAP feature contributions for a True Positive case (Scenario #12). The 3-hour peak forecast provides the strongest positive influence.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

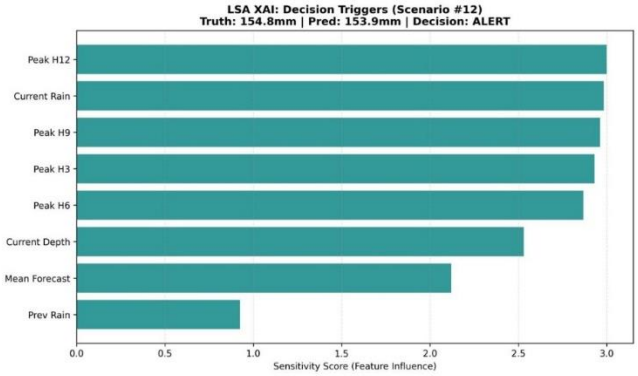

Figure 7: *Local Sensitivity Analysis - True Positive Case Study (Scenario #12)*

The chart shows Peak H12 emerges as the critical trigger, confirming it as the final decision point. The rule-based summary states that the alert is driven by the 3-hour peak forecast. It also indicates that the decision is confirmed because the 12-hour peak forecast is identified as the final tipping point for the alert.

(b). True Negative Case Scenario Study

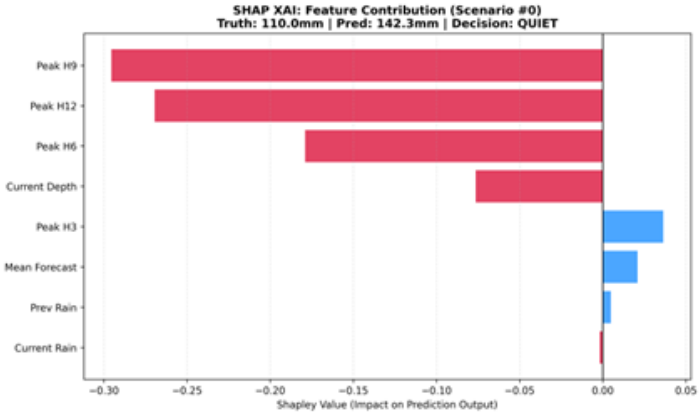

Figure 8: SHAP Feature Contribution - True Negative Case Study (Scenario #0)

The chart shows where negative influences dominated it, particularly from the 9-hour peak forecast, suppressing the alert.

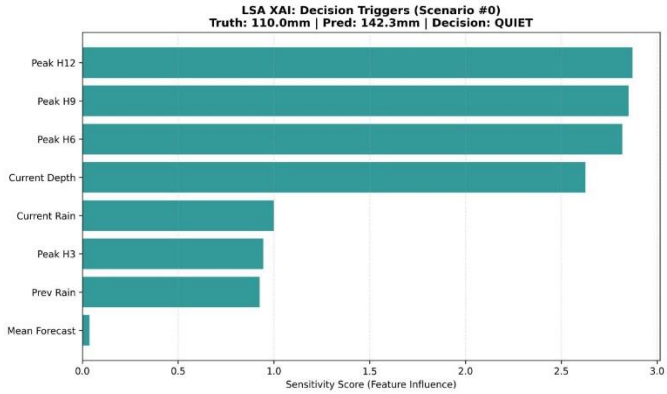

Figure 9: Local Sensitivity Analysis - True Negative Case Study (Scenario #0)

The chart shows Peak H12 shows high sensitivity, indicating high sensitivity and continued monitoring for potential changes.

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

The rule-based summary states that the system remained quiet because the 9-hour peak forecast provided no evidence of imminent risk. It also indicates that the 12-hour peak forecast remained stable within safe limits, confirming no pressure to alert."

D. Performance Comparison and Baseline Study

For performance comparison, a head-to-head test against four "baselines" *(Threshold Rule, Majority Class, Random Forest, and XGBoost)* using the same 382 Houston storm sequences used for evaluation.

Table 6: Complete Baseline Comparison Results (N=382)

Model	Accuracy	Precision	Recall	F1	False Positive Rate (Alert Count)
Threshold Rule (150mm)	0.9921	1.0000	0.9911	0.9955	0.0000 (0/44)
Majority Class	0.8848	0.8848	1.0000	0.9389	1.0000 (44/44)
Random Forest	0.8220	0.9470	0.8462	0.8938	0.3636 (16/44)
XGBoost	0.8010	0.9396	0.8284	0.8805	0.4091 (18/44)
PPO Agent (Proposed)	0.9843	0.9940	0.9882	0.9911	0.0455 (2/44)
Blind PPO (No Forecast)	0.1152	0.0000	0.0000	0.0000	0.0000 (0/44)

Table 6 above shows that the PPO agent performed better than the standard professional models like Random Forest and XGBoost. The Threshold Rule got the highest score, it uses the same rule we used to label the data, the PPO agent reaches 98.43% accuracy it learned to avoid the punishment for making mistakes, mastering the logic on its own. XGBoost and Random Forest triggered false alarms on 36% to 41% of quiet scenarios. They alert heavy rain even if it doesn't cause a flood. The Blind PPO case is used to prove the surrogate is essential, its performance dropped to zero because the agent could not make right decision due to the sensor it was provided not been enough.

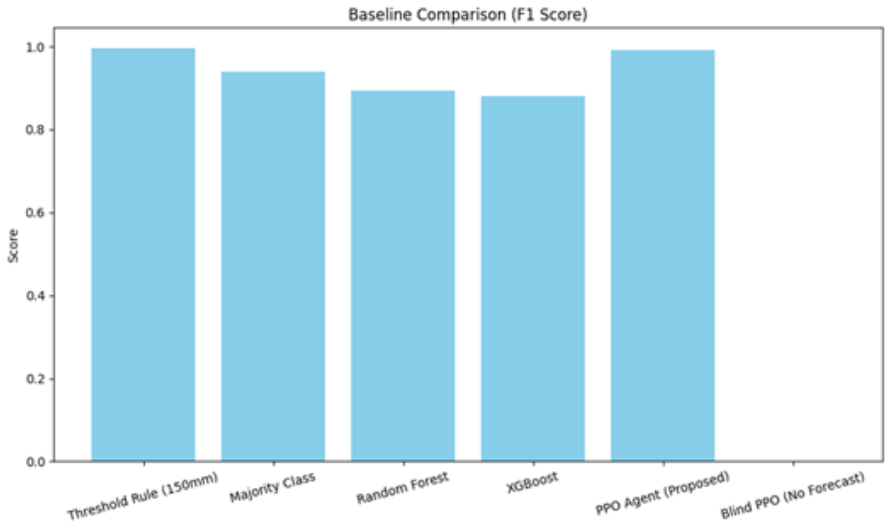

Figure 10: Baseline Comparison of F1 and False Alarm Alert

Volume 2, Issue 2, 2026
https://doi.org/10.5281/zenodo.20388429

E. Quantitative XAI Validation

The XAI explanations are evaluated across 100 storm sequences to ensure the evidence is truthful, with the Top-1 Fidelity score is 0.857. This is calculated by removing the main sensor and noting the drop-in alert confidence, and the 85.7% drop confirms the XAI identifies the exact sensors driving the agent's logic. The Stability score is 0.846, which shows the consistency of the sensor rankings across different datasets. A score of 0.846 proves the reports are reliable and not based on random noise, and these metrics ensure the final reports are built on stable mathematical evidence.

V. CONCLUSION

This study shows that combining fast surrogates with reinforcement learning and explainable AI can form a reliable framework for early urban flood detection. The surrogate achieved a Nash-Sutcliffe Efficiency of 0.9966. That a 96% ability to replicate complex hydrodynamics in milliseconds. This speed enables the reinforcement learning agent to develop decision rules that prioritize public safety during storms. During training reward systems penalize missed floods more heavily than false alarms, guiding the agent to act cautiously in uncertain situations. The trained RL agent achieved an F1-score of 0.9911 and missed only four flood events over the entire period it was trained. SHAP and Local Sensitivity Analysis did provide transparent reasoning for each decision and helped in extraction of rule-based insights that help operators understand and trust the system. In this setup, by preserving static physical inputs while varying the temporal rainfall sequences, the reinforcement learning agent determines decision outcomes, which in turn reveals the environmental factors influencing each case. This setup contributes to timelier and more reliable urban flood detection. The framework relies on open-source data, making the framework practical, scalable, and suitable for different urban contexts, including cities such as Houston. Overall, it offers a working template for addressing high-impact urban climate risks while enabling earlier and more informed flood-related decision-making. Several limitations should be noted. Using global datasets may not fully capture localized variations in urban drainage systems and infrastructure at finer spatial levels. In addition, training the reinforcement learning component can be computationally demanding which may limit applicability in resource-constrained environments. Future work should include validation across multiple geographic regions to evaluate generalizability and real-world performance in municipal settings. Incorporating additional variables, such as coastal surge effects, may further enhance model performance. In summary, this study provides a foundation for applying explainable machine learning methods in a practical and scalable way to support urban flood resilience.

REFERENCES

Mullapudi, A., Lewis, M., Gruden, C., & Kerkez, B. (2020). Deep reinforcement learning for the real-time control of stormwater systems. *Advances in Water Resources, 140*, 103600. doi:10.1016/j.advwatres.2020.103600

Aderyani, F. R., Jafarzadegan, K., & Moradkhani, H. (2025). A surrogate machine learning modeling approach for enhancing the efficiency of urban flood modeling at metropolitan scales. *Sustainable Cities and Society, 123*, 106277. doi:10.1016/j.scs.2025.106277

Choubin, B., Jaafari, A., Henareh, J., Karimi, O., & Hosseini, F. (2025). Explainable artificial intelligence (XAI) for interpreting predictive models and key variables in flood susceptibility. *Results in Engineering, 27*, 105976. doi:10.1016/j.rineng.2025.105976

Roy, B., Goodall, J., McSpadden, D., Goldenberg, S., & Schram, M. (2025). Forecasting Multi-Step-Ahead Street-Scale Nuisance Flooding using a seq2seq LSTM Surrogate

https://doi.org/10.5281/zenodo.20388429

Model for Real-Time Application in a Coastal-Urban City. *Journal of Hydrology, 656*, 132697. doi:10.1016/j.jhydrol.2025.132697

Zeng, B., Huang, G., & Chen, W. (2025). Research progress and prospects of urban flooding simulation: From traditional numerical models to deep learning approaches. *Environmental Modelling & Software, 183*, 106213. doi:10.1016/j.envsoft.2024.106213

Lai, C., Liao, Y., Yu, H., Wang, Z., & Xu, C.-Y. (2025). Formation mechanism analysis and the prediction for compound flood arising from rainstorm and tide using explainable artificial intelligence. *Journal of Environmental Management, 388*, 125858. doi:10.1016/j.jenvman.2025.125858

Dang, T. Q., Tran, B., Le, Q. N., Dang, T. D., Tanim, A. H., Pham, Q. B., et al. (2024). Application of machine learning-based surrogate models for urban flood depth modeling in Ho Chi Minh City, Vietnam. *Applied Soft Computing, 150*, 111031. doi:10.1016/j.asoc.2023.111031

Piadeh, F., Behzadian, K., & Alani, A. (2022). A critical review of real-time modelling of flood forecasting in urban drainage systems. *Journal of Hydrology, 607*, 127476. doi:10.1016/j.jhydrol.2022.127476

Qin, H., Liang, Q., Chen, H., & De Silva, V. (2024). A Coupled Human and Natural Systems (CHANS) framework integrated with reinforcement learning for urban flood mitigation. *Journal of Hydrology, 643*, 131918. doi:10.1016/j.jhydrol.2024.131918

He, Z., Tian, W., Wang, J., Yan, H., Xin, K., & Tao, T. (2025). Deep reinforcement learning control as an innovative approach for urban drainage systems: review and prospects. *Water Research, 284*, 123954. doi:10.1016/j.watres.2025.123954

Goyal, H. R., Ghanshala, K. K., & Sharma, S. (2021). Flash flood risk management modeling in Indian cities using IoT based reinforcement learning. *Materials Today: Proceedings, 46*, 10533–10538. doi:10.1016/j.matpr.2021.01.072

Li, W., Liu, C., Xu, Y., Niu, C., Li, R., Li, M., et al. (2024). An interpretable hybrid deep learning model for flood forecasting based on Transformer and LSTM. *Journal of Hydrology: Regional Studies, 54*, 101873. doi:10.1016/j.ejrh.2024.101873

Li, X., Liang, X., Wang, X., Wang, R., Shu, L., & Xu, W. (2023). Deep reinforcement learning for optimal rescue path planning in uncertain and complex urban pluvial flood scenarios. *Applied Soft Computing, 144*, 110543. doi:10.1016/j.asoc.2023.110543

Liao, Y. W. (2023). Fast simulation and prediction of urban pluvial floods using a deep convolutional neural network model. *Journal of Hydrology, 624*, 129945. doi:10.1016/j.jhydrol.2023.129945

Xu, L. (2024). A hybrid surrogate model for real-time coastal urban flood prediction: An application to Macao. *Journal of Hydrology, 642*, 131863. doi:10.1016/j.jhydrol.2024.131863

McSpadden, D., Goldenberg, S., Roy, B., Schram, M., Goodall, J., & Richter, H. (2024). A comparison of machine learning surrogate models of street-scale flooding in Norfolk, Virginia. *Machine Learning with Applications, 15*, 100518. doi:10.1016/j.mlwa.2023.100518

Pradhan, B., Lee, S., Dikshit, A., & Kim, H. (2023). Spatial flood susceptibility mapping using an explainable artificial intelligence (XAI) model. *Geoscience Frontiers, 14*, 101625. doi:10.1016/j.gsf.2023.101625

Qin, R., Khakzad, N., & Zhu, J. (2020). An overview of the impact of Hurricane Harvey on chemical and process facilities in Texas. *International Journal of Disaster Risk Reduction, 45*, 101453. doi:10.1016/j.ijdrr.2019.101453

Ren, H., Pang, B., Zhao, G., Liu, Y., Zhang, H., & Liu, S. (2025). Rapid flood simulation and source area identification in urban environments via interpretable deep learning. *Journal of Hydrology, 651*, 132551. doi:10.1016/j.jhydrol.2024.132551

Tian, W., Xin, K., Zhang, Z., Zhao, M., & Liao, Z. (2023). Flooding mitigation through safe & trustworthy reinforcement learning in urban drainage systems. *Journal of Hydrology, 620*, 129435. doi:10.1016/j.jhydrol.2023.129435

Chu, W., Zhang, C., Li, H., Zhang, L., Shen, D., & Li, R. (2024). SHAP-powered insights into spatiotemporal effects: Unlocking explainable Bayesian-neural-network urban flood forecasting. *International Journal of Applied Earth Observation and Geoinformation, 131*, 103972. doi:10.1016/j.jag.2024.103972

Lei, X., Chen, W., Panahi, M., Falah, F., Rahmati, O., Uuemaa, E., et al. (2021). Urban flood modeling using deep-learning approaches in Seoul, South Korea. *Journal of Hydrology, 601*, 126684. doi:10.1016/j.jhydrol.2021.126684

Fang, Z., Wang, Y., Peng, L., & Hong, H. (2021). Predicting flood susceptibility using LSTM neural networks. *Journal of Hydrology, 594*, 125734. doi:10.1016/j.jhydrol.2020.125734

Li, Z., & Zhang, Z. (2025). Artificial intelligence-incorporated prediction for urban flooding processes in the past 20 years: A critical review. *Environmental Modelling & Software, 192*, 106525. doi:10.1016/j.envsoft.2025.106525

Zhang, Z., Lu, C., Tian, W., Liao, Z., & Yuan, Z. (2024). Graph neural network-based surrogate modelling for real-time hydraulic prediction of urban drainage networks. *Water Research, 263*, 122142. doi:10.1016/j.watres.2024.122142

https://doi.org/10.5281/zenodo.20388429