Environmental Data Science (2023), **xx**:xx 1–19

CAMBRIDGE
UNIVERSITY PRESS

Article Type: Applications paper

FloodGNN-GRU: A Spatio-Temporal Graph Neural Network for Flood Prediction

Arnold Kazadi[1], James Doss-Gollin[2], Antonia Sebastian[3] and Arlei Silva*[1]

[1]Department of Computer Science, Rice University, Houston, 77005, Texas, USA * E-mail: arlei@rice.edu.
[2]Department of Civil and Environmental Engineering, Rice University, Houston, 77005, Texas, USA.
[3]Department of Earth, Marine and Environmental Sciences, University of North Carolina at Chapel Hill, Chapel Hill, 27599, North Carolina, USA.

Keywords: flood prediction, machine learning, deep learning, graph neural networks

Abstract

Classical approaches for flood prediction apply numerical methods for the solution of partial differential equations that capture the physics of inundation processes (e.g. the 2D Shallow Water equations). However, traditional inundation models are still unable to satisfy the requirements of many relevant applications, including early-warning systems, high-resolution (or large spatial domain) simulations, and robust inference over distributions of inputs (e.g. rainfall events). Machine learning (ML) approaches are a promising alternative to physics-based models due to their ability to efficiently capture correlations between relevant inputs and outputs in a data-driven fashion. In particular, once trained, ML models can be tested/deployed much more efficiently than classical approaches. Yet, few ML-based solutions for spatio-temporal flood prediction have been developed and their reliability/accuracy is poorly understood. In this paper, we propose FloodGNN-GRU, a spatio-temporal flood prediction model that combines a graph neural network (GNN) and a Gated Recurrent Unit (GRU) architecture. Compared to existing approaches, FloodGNN-GRU (i) employs a graph-based model (GNN); (ii) operates on both spatial and temporal dimensions; and (iii) processes the water flow velocities as vector features, instead of scalar features. We evaluate FloodGNN-GRU using a LISFLOOD-FP simulation of Hurricane Harvey (2017) in Houston, Texas. Our results, based on several metrics, show that FloodGNN-GRU outperforms several data-driven alternatives in terms of accuracy. Moreover, our approach can be trained 100x faster and tested 1000x faster than the time required to run a comparable simulation. These findings illustrate the potential of ML-based methods to efficiently emulate physics-based inundation models, especially for short-term predictions.

Impact Statement

A spatio-temporal model for flood prediction combining a graph neural network and a gated recursive unit can generate accurate short-term predictions for a simulation of Hurricane Harvey in Houston, TX, while requiring significantly less computation time.

1. Introduction

Flooding is the most devastating type of natural disaster both socially and economically [1]. Climate change is driving changes in the intensity, frequency, and spatiotemporal structure of heavy precipitation, which is anticipated to increase urban flood hazard in many regions [2, 3]. Predictive modeling can support adaptation in many ways, such as through early warning systems or by mapping hazards across space and time [1].

The theoretical framework for flood modeling is based on fluid mechanics, such as the 3D Navier Stokes equations. Due to numerical constraints, insufficient knowledge of boundary conditions, and

limited computational resources, most state-of-the-art models leverage sophisticated numerical methods to apply 1D and 2D equations to model flood propagation. Despite these simplifications, computational cost remains a critical bottleneck. For example, LISFLOOD-FP [4]—a popular open-source flood simulation package—takes approximately 6 hours to simulate Hurricane Harvey with a 1-hour time resolution and 30 meters spatial resolution over one week for the Harris County region (approximately 5,000 km^2) in Texas using a desktop computer, even though LISFLOOD-FP uses a simplified physical scheme. This high computational cost limits our ability to run large flood ensembles (e.g., to evaluate different mitigation strategies over large samples of rainfall events), to deploy early warning systems, or to increase the spatio-temporal resolution and domain of simulations.

Recently, machine learning (ML) has been presented as an alternative to physics-based inundation models [5]. For instance, ML has been applied for real-time flood forecasting [6], continental-scale flood risk assessment [7, 8], high-resolution flood extent prediction [9], and resource-constrained prediction [10]. Many of these approaches apply deep learning due to its expressive power [11] and scalability. However, these models either focus on the spatial or the temporal dimension [2, 12, 13, 14, 15, 16, 17, 18, 19], which limits out-of-sample predictive skill. More specifically, spatial models, which are based on Convolutional Neural Networks (CNNs) or feed-forward neural networks, predict only the maximum water depth at each location (a.k.a., an inundation map). On the other hand, temporal models apply Recurrent Neural Networks (RNNs) to capture the evolution of water depths over time without accounting for the spatial structure [20, 21, 22, 23, 24]. These restrictions limit the applicability of ML-based flood prediction approaches compared to more traditional physics-based solutions. For instance, while capturing the dynamics of the flooding event is necessary to recommend evacuation routes, spatial information enables the prediction of physically consistent water depths based on conservation laws. ML models are often trained with either (sparse) gauge observations or with (dense) outputs of physics-based flood simulators. Their major advantage is fast test/deployment time compared to state-of-the-art physics-based simulators. On the other hand, different from physics-based models, fully data-driven approaches do not encode fluid mechanics equations, thus only being able to learn the physics of flooding directly from data.

This work investigates data-driven spatiotemporal models for flood prediction. We focus on graph-based models (allowing irregular mesh), where the raster map of a region is represented as nodes/cells as locations and edges as spatial proximity. Graph Neural Networks, which can be viewed as a generalization of CNNs, have achieved promising results in predicting physics simulations [25]. Graphs are more flexible than image-based representations, as they support irregularly-sampled cells and rotation-invariance, while still being able to capture relations between nearby locations. We leverage these advantages by proposing FloodGNN-GRU, a Graph Neural Network architecture for flood prediction. At each time step, FloodGNN-GRU predicts the water depths and velocities—i.e. the state of the flood—based on previous depths and velocities as well as static features (e.g., elevation). Velocities are processed as vector features using geometric vector perceptrons [26]. We validate our model in terms of accuracy and computation time using a LISFLOOD-FP simulation of Hurricane Harvey, in Houston, TX. Our experiments illustrate the potential of FloodGNN-GRU to be a faster alternative to traditional physics-based flooding simulation schemes. The main contributions of our work can be summarized as follows:

- We propose FloodGNN-GRU, a graph neural network designed for flood prediction;
- We propose the use of geometric vector perceptrons to represent velocity information in FloodGNN-GRU and show its improvement in performance;
- We evaluate FloodGNN-GRU using a representative dataset that simulates Hurricane Harvey in Houston, TX using LISFLOOD-FP. Our experiments show that our approach outperforms the best baseline by 17% in terms of RMSE and 31% in terms of Pearson's coefficient of correlation. Moreover, once trained, FloodGNN-GRU is faster than LISFLOOD-FP at testing/deployment (1000x faster).

2. Related Work

There is a vast literature on traditional inundation modeling [1, 27]. The most physically accurate mathematical inundation model is the 3-D Navier Stokes equation at the resolution of millimeters. Multiple limitations make such a model impractical, including computation and unavailability of high-resolution data. Instead, existing flood simulation models apply more tractable alternatives, such as a combination of 1-D and 2-D Saint-Venant (or shallow water) equations at the resolution of meters [28, 29]—with some variation depending on whether the covered area is urban or rural. These simplified models are implemented using sophisticated high-performance schemes to enable efficient computation using large-scale distributed systems and specialized hardware. Recent advances towards each of these directions have enabled the application of inundation models at continental and global scales [7, 8, 1] However, this is still an area of active research towards the development of higher (spatial and temporal) resolution, larger scale, and more accessible inundation prediction.

Recently, machine learning, and especially deep learning, has gained popularity as an alternative to traditional inundation models. For instance, in [10], the authors describe an operational data-driven system that integrates a long short-term memory (LSTM) architecture, thresholding, and a manifold model for flood forecasting. Kao et al [30] address the same problem using a combination of an LSTM and an autoencoder. A feed-forward neural network for high-resolution flood forecasting (4 meters) is introduced in [31]. Random forests have been applied to predict flood hazards at the scale of the continental United States in [7] and [8]. The problem of maximum inundation prediction has been addressed using an ensemble of neural networks [2], CNNs [19, 18], and a U-Net [32]. In [17], inundation maps are generated using an adversarial network conditioned on the rainfall distribution. Mosavi et al. [5] provides a comprehensive review of machine learning approaches for flood prediction. Bentivoglio et al. [33] reviews machine learning applications for flood mapping (e.g. inundation maps, inundation hazard maps). Similar to [31], our paper is focused on predicting water depths (and velocities) at multiple steps using machine learning. However, we investigate the use of graph neural networks with geometric feature representations as a more effective architecture for flood prediction.

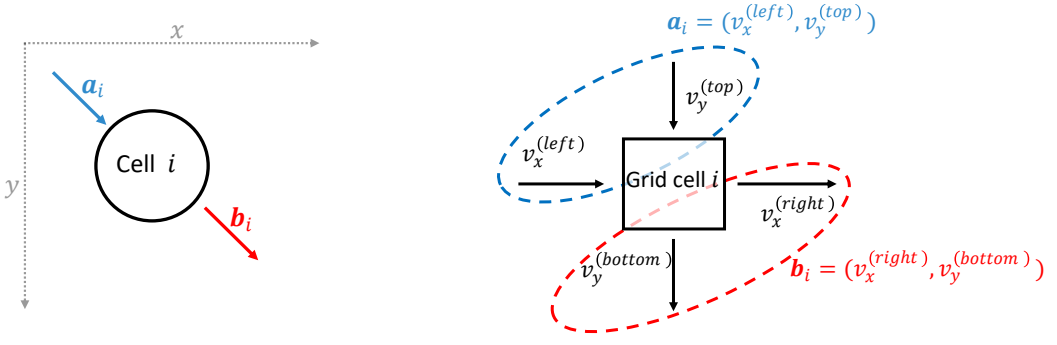
One of the key challenges in applying data-driven models to scientific computing applications is the lack of physical knowledge encoded in the model. As a consequence, a machine learning model can generate predictions that violate well-known physical laws (e.g. conservation of mass). To address this challenge, recent papers have proposed incorporating physics into deep learning models via physics-inspired neural networks (PINNs) [34, 35]. The idea is to add a differential equation capturing the physics of the system to the loss function of the machine learning model. As a consequence, predictions that violate the known physical relationships between the variables are penalized.

As flooding is a spatio-temporal process, reproducing the physics of flooding requires a spatial model. Here, we apply Graph Neural Networks (GNNs) to capture spatial information. GNNs enable the application of deep learning to irregular graph data. [36, 37, 38, 39, 40, 41]. GNNs have been successfully applied to several problems including node classification, link prediction and graph classification. The inference is often performed via message-passing among vertices in the graph [41], which can be performed efficiently via sampling [40]. More recently, there has been a growing interest in applying GNNs for physics-based simulations [42, 43, 44, 45, 46, 47]. For instance, in [48] the authors propose a GNN that can simulate dynamics of fluids, rigid solids, and deformable materials via message-passing. A mesh-based GNN for physical simulations with adaptive re-meshing is introduced in [25]. Two GNNs for weather forecasting were recently shown to achieve promising results in [49] and [50].

In this paper, we apply GNNs for flood prediction. Preliminary results for this paper were presented in [51], where we introduced FloodGNN. Here, we extend FloodGNN to account for spatially distributed rainfall data and provide additional experiments validating our approach. A GNN for flood prediction was also proposed in [52]. However, notice that the previous work does not account for rainfall data and is evaluated using completely synthetic datasets. Moreover, our method represents water velocity in its physical form as a vector feature, instead of scalars. Our experiments apply data from a more realistic simulation of Hurricane Harvey in Houston, Texas, generated using LISFLOOD-FP.

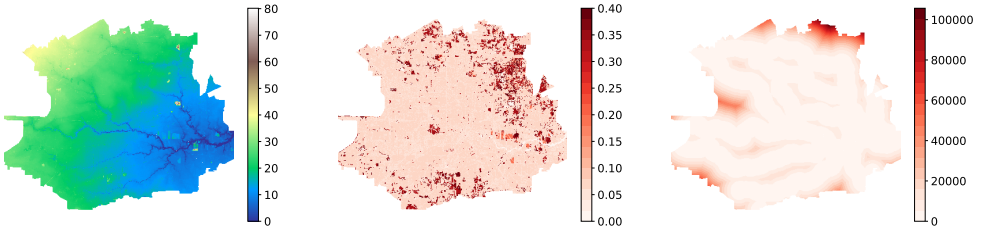
3. Flood prediction problem

We will first formalize the problem investigated in this work. Without loss of generality, we will assume that the locations are organized as graphs; for instance, a mesh grid where cells are considered as nodes and edges connect adjacent cells. The goal is to predict water depths w_i^{t+1} and in/out-velocity vectors $\mathbf{a}_i^{t+1}$ ($\mathbf{b}_i^{t+1}$) (see Figure 1) on each node i based on attributes capturing the topography of the locations, rainfall predictions, and past values of water depth $w_i^{t-k} \ldots, w_i^{t-1}, w_i^t$ and velocity $(\mathbf{a}_i^{t-k}, \mathbf{b}_i^{t-k}), \ldots, (\mathbf{a}_i^{t-1}, \mathbf{b}_i^{t-1}), (\mathbf{a}_i^t, \mathbf{b}_i^t)$. We consider the ground elevation e_i, the Manning friction coefficient n_i, and the distance to the closest river d_i as static topographic attributes (see Figure 2). Moreover, rainfall predictions p_i^t, which will impact predicted depths and velocities, are given for each location. We note that our model is general enough to account for other static features, including those derived from the set we have considered (e.g., slope and aspect). We do not consider features with spatial information such as the absolute location of nodes, which can violate the rotation-invariance property of the graph representation. Our goal is to provide a compact set of features to the model and allow it to learn more complex (composite) features directly from data.



(a) General in-/out- velocity depiction. (b) In-/out-velocity on a staggered grid.

Figure 1: **(1a)** On a general graph, each node/cell i has an inflow with an *in*-velocity $\mathbf{a}_i$ and outflow with an *out*-velocity $\mathbf{b}_i$. We note that water can enter and exit the cell in any direction and $\mathbf{a}_i$ and $\mathbf{b}_i$ will differ depending on the properties of node i (e.g., friction and elevation). **(1b)** the same vectors can be represented on a staggered grid using (left and top) cell interfaces as a basis for *in*-velocity $\mathbf{a}_i$ and (right and bottom) cell interfaces as a basis for *out*-velocity. This convention is applied by LISFLOOD-FP [53].



(a) Ground elevation. (b) Manning's friction (c) Distance to closest stream

Figure 2: Spatial distribution of static features in our dataset.

4. Problem formulation and approach

FloodGNN-GRU is a spatio-temporal model that combines a graph neural network (GNN), named FloodGNN, and a gated recurrent unit (GRU) network [54]. Its predictions are based on D-dimensional dynamic. vector representations $H^{t+1} \in \mathbb{R}^{n \times D}$ as $H^{t+1} = \text{FloodGNN-GRU}(H^t, X^t, G)$, where n is the number of nodes, $H^t \in \mathbb{R}^{n \times D}$ are latent representations, used as hidden states in the GRU module (at time t), X^t combines node scalar and vector attributes at time t, G is the graph topology (i.e. set of nodes and edges), and D is a hyperparameter.

4.1. Formulation

We represent the flood dynamics within a grid with states $R_g^1, \dots R_g^T$, where the topology remains constant by the node features change over time. At time t, the system is in state R_g^t where the nodes (i.e., grid cells) $v_i \in V$ are associated with vector features $\mathbf{V}_i^t$ and scalar features $\mathbf{s}_i$. As vector features, we consider $\mathbf{V}_i^t = \left[\mathbf{a}_i^t / \|\mathbf{a}_i^t\|, \mathbf{b}_i^t / \|\mathbf{b}_i^t\|\right]^T \in \mathbb{R}^{2\times 2}$, which are *in-* and *out-* velocities (See Figure 1). As scalar features, we consider $\mathbf{s}_i^t = (e_i, n_i, d_i, \|\mathbf{a}_i^t\|, \|\mathbf{b}_i^t\|, w_i^t)^T \in \mathbb{R}^6$ (see problem definition in Section 3). Our goal is, given the current state R_g^t of R_g, to predict the depth w_i^{t+1} and velocity $\mathbf{V}_i^{t+1}$ for each node $v_i \in V$ at time step $t+1$.

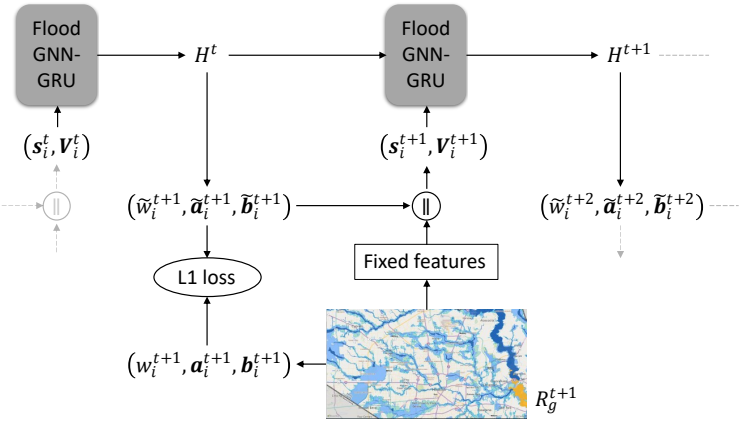


Figure 3: **Overview of FloodGNN-GRU.** At each time t, the region R_g is in state R_g^t with scalar features $\mathbf{s}_i^t$ and vector features $\mathbf{V}_i^t$ for each node/cell v_i. These are processed through a FloodGNN-GRU to produce hidden state H^t that captures both spatial and temporal information on the dynamics of a flooding event. H^t is later used for the estimation of the next water depth $\tilde{w}_i^{t+1}$ and velocities $\tilde{\mathbf{a}}_i^{t+1}$ and $\tilde{\mathbf{b}}_i^{t+1}$. The L1 loss function between $\tilde{w}_i^{t+1}$, $\tilde{\mathbf{a}}_i^{t+1}$, $\tilde{\mathbf{b}}_i^{t+1}$ and their ground truth values w_i^{t+1}, $\mathbf{a}_i^{t+1}$, $\mathbf{b}_i^{t+1}$ is used for parameter learning in our model.

4.2. Method

In FloodGNN-GRU, a graph neural network (FloodGNN) captures the spatial behavior of a flooding event as it spreads over a region. This is performed using the message-passing mechanism proposed in [41]. The temporal evolution of the flood is captured by a gated recurrent unit (GRU), which is a type of Recurrent Neural Network [55, 56]. These two components of our proposed method are introduced in the three sections following sections.

4.2.1. Spatial information with FloodGNN

FloodGNN represents a given area as a graph by sampling nodes over the space. Topographic and rainfall data from the corresponding area are assigned to each node, and nodes are connected based on spatial adjacency. As velocities are vectors, we would like to preserve their geometry and not treat them as scalar features. Thus, we apply geometric vector perceptrons (GVP) [26] for feature transformation with attention mechanism during message passing as proposed by [57]. GVPs are an extension of standard dense layers (MLPs) that consider two types of features, scalar features ($\mathbf{s} \in \mathbb{R}^C$) and vector features ($\mathbf{V} \in \mathbb{R}^{D\times 2}$). The former is the concatenation of features that by nature are c scalars (e.g., ground elevation) and the latter is a collection of d features that have some physical/geometric properties and are therefore represented as vectors (e.g., velocity in 2D). GVP takes a tuple of scalar features and vector features $(\mathbf{s}, \mathbf{V})$ to produce a new tuple of scalar and vector features $(\mathbf{s}' \in \mathbb{R}^M, \mathbf{V}' \in \mathbb{R}^{N\times 2})$.

Thus, $(\mathbf{s}', \mathbf{V}') = \text{GVP}(\mathbf{s}, \mathbf{V})$, slightly modified from [26], is defined as follows:

$$
\begin{aligned}
\mathbf{V}^{(h)} &= \mathbf{W}_h \mathbf{V} \in \mathbb{R}^{F\times 2} \\
\mathbf{V}' &= \mathbf{W}\mathbf{V}^{(h)} \in \mathbb{R}^{N\times 2} \\
\mathbf{r} &= \|\mathbf{V}^{(h)}\|_2 \in \mathbb{R}^F \text{ (row-wise L2-norm)} \\
\mathbf{q} &= [\mathbf{r} \parallel \mathbf{s}] \in \mathbb{R}^{F+C} \text{ (concatenation)} \\
\mathbf{s}' &= \mathbf{W}_s \mathbf{q} + \mathbf{b} \in \mathbb{R}^M
\end{aligned}
$$

where $\mathbf{W}_h \in \mathbb{R}^{F\times D}, \mathbf{W} \in \mathbb{R}^{N\times F}, \mathbf{W}_s \in \mathbb{R}^{M\times(F+C)}$, and $\mathbf{b} \in \mathbb{R}^M$ are learnable weights. Vector operations (multiplication by a weight matrix) preserve equivariance to vector operations (combinations of rotations and reflections) while the scalar operations are more expressive (vector norms are treated as scalars).

FloodGNN is therefore similar to traditional GNNs but replaces their simple multiplication by weight matrices (in message passing and update operations) with GVP operations Given a node v_i with immediate neighborhood $\mathcal{N}(v_i) = \{v_i\} \cup \{v_j : \text{there is an edge } v_j \to v_i\}$, the node update operation is performed as follows

$$
\begin{aligned}
(\mathbf{p}_j, \mathbf{P}_j) &= \text{GVP}(\mathbf{s}_j, \mathbf{V}_j) \qquad \forall v_j \in \mathcal{N}(v_i) \\
\mathbf{s}_i' &= \sum_{j|v_j\in\mathcal{N}(v_i)} \alpha_j \mathbf{p}_j, \qquad \mathbf{V}_i' = \sum_{j|v_j\in\mathcal{N}(v_i)} \beta_j \mathbf{P}_j
\end{aligned}
$$

The scalars α_j and β_j are attention weights assigned to v_j when passing message to v_i [58]:

$$
\alpha_j = \frac{\exp(\langle \mathbf{p}_i, \mathbf{p}_j \rangle)}{\sum_{k|v_k\in\mathcal{N}_{(v_i)}} \exp(\langle \mathbf{p}_i, \mathbf{p}_k \rangle)}, \qquad \beta_j = \frac{\exp(\text{tr}(\mathbf{P}_i^T \mathbf{P}_j))}{\sum_{k|v_k\in\mathcal{N}_{(v_i)}} \exp(\text{tr}(\mathbf{P}_i^T \mathbf{P}_k))}
$$

where $\langle\cdot\rangle$ is the inner product between two vectors and $tr(\cdot)$ is the trace of a matrix:

During training, FloodGNN. learns to generate representations $(\mathbf{s}', \mathbf{V}')$ that capture the state of the flooding event (i.e. water depths and velocity vectors). In the next section, we describe how a GRU can be combined with FloodGNN to enable it to capture the dynamics of the flooding event.

4.2.2. Spatio-temporal information with GRU and FloodGNN

FloodGNN-GRU applies a GRU model to capture the temporal information from the flooding event. The MLP module from the (traditional) GRU is replaced with FloodGNN layers to leverage the temporal and spatial spread of a flood. Our approach follows a similar strategy to the ConvLSTM architecture [59] to combine spatial and temporal information, but FloodGNN plays the role of the Convolutional Neural Network. This enables our approach to process a sequence of relevant node inputs (topographic attributes, rainfall, previous water depths, etc.) to predict the next state of the flooding event (water depths and velocity vectors).

The GRU is designed as follows:

$$z^t = \text{sigmoid}\left(f_z(R_g^t) + g_z(H^{t-1})\right) \tag{4.1}$$

$$r^t = \text{sigmoid}\left(f_r(R_g^t) + g_r(H^{t-1})\right) \tag{4.2}$$

$$\hat{H}^t = \tanh\left(f_h(R_g^t) + g_h(r_t \odot H^{t-1})\right) \tag{4.3}$$

$$H^t = z^t \odot H^{t-1} + (1 - z^t) \odot \hat{H}^t \tag{4.4}$$

where $f_{\{z,r,h\}}$ and $g_{\{z,r,h\}}$ are FloodGNN layers (see Section 4.2.1) and *sigmoid* and *tanh* are non-linear activation functions. R_g^t is the state of the graph at time t, that is, the set of tuples of scalar features and vector features of all the nodes in the graph at time step t. H^{t-1} is the set of tuples of scalar hidden states and vector hidden states from the previous time step $t-1$. Note that all these operations have inputs and output sets as tuples (of scalar and vector features). That is, $z^t, r^t, \hat{H}^t$, and H^t are all tuples We use $\odot$ as the element-wise product between vectors or matrices. Here again, since we are dealing with tuples, $\odot$ is applied individually/separately to scalar and vector parts of r^t, H^{t-1}, and z^t. Furthermore, all operations, including activation functions and arithmetic operations, are applied individually to the entries of these tuples

4.2.3. Prediction

The values of the water depth $t+1$ ($\tilde{w}_i^{t+1}$), and associated velocities ($\tilde{\mathbf{a}}_i^{t+1}$, and $\tilde{\mathbf{b}}_i^{t+1}$) at the next time step are predicted based on node representations as follows:

$$(s, \mathbf{V}) = f_p(H^t), \qquad s \in \mathbb{R} \quad \mathbf{V} \in \mathbb{R}^{2\times 2}$$
$$\tilde{w}_i^{t+1} = s^2, \quad \tilde{\mathbf{a}}_i^{t+1} = \mathbf{V}_{[1,:]}, \quad \tilde{\mathbf{b}}_i^{t+1} = \mathbf{V}_{[2,:]}$$

where f_p is a GVP layer, H^t is defined in Equation 4.4, $\tilde{\mathbf{a}}_i^{t+1}$ and $\tilde{\mathbf{b}}_i^{t+1}$ are the first row ($\mathbf{V}_{[1,:]}$) and second row ($\mathbf{V}_{[2,:]}$) of $\mathbf{P}$, respectively. We take the square of s to obtain $\tilde{w}_i^{t+1}$ to enforce water depths to always be non-negative.

The values $\tilde{w}_i^{t+1}$, $\tilde{\mathbf{a}}_i^{t+1}$ and $\tilde{\mathbf{b}}_i^{t+1}$ are used to construct input features $\mathbf{s}_i^{t+1}$ and $\mathbf{V}_i^{t+1}$, which are used together with H_t at the next time-step $(t+2)$. The L1 loss, which performed better than L2 loss (see Figure 10), is used to compare predictions $\tilde{w}_i^{t+1}$, $\tilde{\mathbf{a}}_i^{t+1}$, $\mathbf{b}_i^{t+1}$ and their respective ground truth values w_i^{t+1}, $\mathbf{a}_i^{t+1}$, $\mathbf{b}_i^{t+1}$ to update the model parameters. By minimizing the loss for a training set, we optimize both the FloodGNN and GRU parameters in an end-to-end fashion. Once the model is trained, predictions can be made efficiently based on forward operations.

The overall architecture of FloodGNN-GRU is shown in Figure 3. FloodGNN-GRU can predict the dynamics of flooding events based on both topographic attributes and rainfall forecasts. In particular, the recursive nature of our model, inherited from the GRU architecture, enables it to make predictions with long-term lead times—i.e. number of time steps in the future. In the next section, we will evaluate our model using a representative dataset from Hurricane Harvey, in Houston, TX.

5. Experiments

This section provides an empirical evaluation of FloodGNN-GRU, which is our data-driven approach for flood prediction that combines a GNN and a GRU architecture. The goal of our evaluation is to address the following questions: (1) how accurate is FloodGNN-GRU at predicting water depths for various lead times compared to alternative data-driven approaches and (2) how efficiently can FloodGNN-GRU be trained and tested compared to a traditional physics-based inundation model?

5.1. Dataset

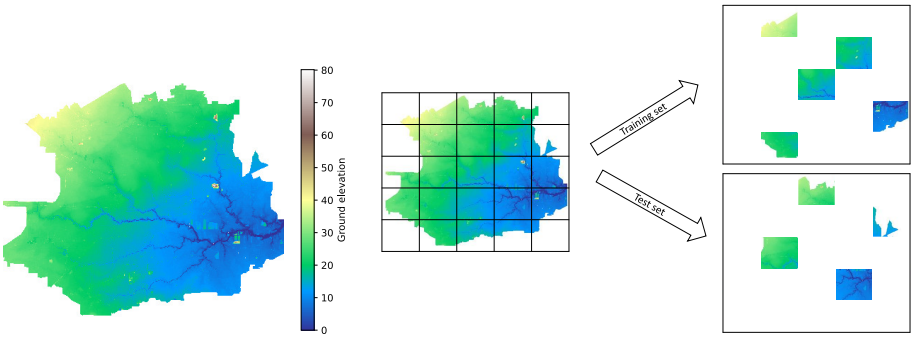
Our experiments are based on simulations from the grid-based flood inundation model LISFLOOD-FP (version 8) [4]. The model domain was designated using a modified shapefile of the Harris County Flood Control District's (HCFCD) watershed boundaries as a mask. Elevation data were interpolated onto the model grid using a 10-m digital elevation model (DEM) collected from the U.S. Geological Survey's (USGS) 3D Elevation Program (3DEP) National Elevation Dataset (NED). River bathymetry data (i.e., bed elevation, bank elevation, channel width) were extracted from HEC-RAS models maintained by HCFCD and available via the Model and Map Management (M3) System and used to create the sub-grid channels (SGC) embedded within the model domain. Overland roughness coefficients were assigned to each grid cell based on the land use/land cover (LULC) classes in the 2016 Multi-Resolution Land Characteristics (MLRC) Consortium's National Land Cover Database (NLCD), which classifies 16 land cover types at a 30-m resolution for the CONUS. Manning's friction coefficients were obtained for each LULC class from Kalyanapu et al. (2009). Infiltration was assumed to be spatially uniform across the model domain (1.5E-6 ms^{-1}). The final 30-m resolution model contains 3,208,196 (1961×1636) active grid cells and encompasses a large portion of Harris County and the City of Houston (Figure 4a).

To validate the LISFLOOD-FP model performance, we hindcast Hurricane Harvey for 6 days beginning on August 25, 2017 00:00 UTC. Runoff processes were simulated by forcing the model with observed precipitation and streamflow. Hourly NOAA NEXRAD radar precipitation records were obtained from the Multi-Radar Multi-Sensor Gauge Corrected (MRMS-GC) Quantitative Precipitation Estimation (QPE) product [60][1]. Upstream boundary conditions were included at the outlets of Addicks and Barker reservoirs using reported hydrographs at USGS streamflow gages 08072600 and 08073100 and at the eastern boundary of the model domain using observed water level records at the NOAA tide gage located at Manchester (Station ID: 8770777). The simulation was run using an adaptive time stepping algorithm based on the convergence condition by Courant–Friedrichs–Lewy to ensure stability and convergence and gridded water levels were saved every 3600 seconds during the simulation. We compare modeled and observed water levels at 73 USGS high water marks (HWMs) and calculate a root-mean-square-error (RMSE) of 1.07, bias (also known as mean error) of 0.82, and R2 of 0.98.

Using the validated model, we obtained hourly water depths and velocity vectors (i.e., time series with a time-step of 1 hour). The model domain was divided into smaller non-overlapping sub-regions of sizes $\approx 50 \times 32$ to generate enough sample regions for training (illustrated in Figure 4b) the baselines and our proposed model. The models are trained and tested on different sub-regions to simulate the setting where our model is applied as an alternative LISFLOOD-FP. The simulation outputs from LISFLOOD-FP are, thus, considered as ground truth or true water depths/velocities for training as well as testing in our work. We obtained the graph representations of these regions by considering a grid cell as a node that is linked to its surrounding cells. There were 1531 grid-based (non-overlapping) sub-regions from which we randomly selected 70% for training, 15% for validation, and 15% for testing. We *randomly* split the data to avoid having in one split, say, training split, regions with one predominantly common feature (for instance, all flat areas); we further consider 3 random independent splits, resulting in 3 experiment results (See Section 5.3).

The generated flood inundation data is heavily left-skewed, i.e., a majority of water depths are zero. This causes some challenges to the training process of a machine learning model such as back-propagation of gradient zero. Therefore, to remove the excessive number of (leading) zeroes from the data, for each sample sub-region, we start counting the first time step where there is at least one grid cell with a non-zero water depth value. Table 1 shows features used in our experiments and the normalization employed for each one of them. In most cases, feature values are scaled down by their order of magnitude. For instance, rainfall values can be of 2 orders of magnitude; thus, they are multiplied by 1e-2.

[1] https://github.com/dossgollin-lab/climate-data/



(a) (b)

Figure 4: (4a) Digital elevation model (DEM) of Houston, Texas. (4b) Generation of non-overlapping sub-regions and illustrative example of the split of these sub-regions into training and test sets

Table 1: Input features. Pre-processing means that the data were calculated from other features.

Feature	Source	Normalization
Digital Elevation Model (e_i)	U.S. Geological Survey	scaled by 1e-2
Distance to closest stream (d_i)	Pre-processing	scaled by 1e-5
Manning's coefficient of friction (n_i)	Kalyanapu et al. [61]	None
In-velocity ($\mathbf{a}_i$)	LISFLOOD-FP	unit vector
Out-velocity ($\mathbf{b}_i$)	LISFLOOD-FP	unit vector
In-velocity magnitude ($\|\mathbf{a}_i\|$)	Pre-processing	scaled by 1e-2
Out-velocity magnitude ($\|\mathbf{b}_i\|$)	Pre-processing	scaled by 1e-2
Rainfall (p_i)	NOAA NEXRAD	scaled by 1e-2
Water depth (w_i)	LISFLOOD-FP	scaled by 1e-1

5.2. Evaluation metrics

We used metrics commonly employed in regression tasks for the evaluation of and comparison between the performances of our proposed method and baselines. In the notation used below, y_i is the true water depth of a cell, p_i is the predicted water depth of the cell, $\bar{y}_i$ is the mean of water depth over all the cells, and N is the total number of all the cells (from all the sub-regions combined). While comparing the methods based on a single evaluation metric might not be sufficient, this comprehensive set of metrics, in combination, provides a clear picture of their performance.

- Root mean square error:

$$\text{RMSE} = \sqrt{\frac{1}{N}\|y_i - p_i\|_2^2}$$

- Nash–Sutcliffe model efficiency coefficient:

$$\text{NSE} = 1 - \frac{\sum_i^N \|y_i - p_i\|_2^2}{\sum_i^N \|y_i - \bar{y}_i\|_2^2}$$

- Pearson correlation coefficient:

$$r = \frac{\sum_i^N (y_i - \bar{y}_i)(p_i - \bar{p}_i)}{\sqrt{\sum_i^N (y_i - \bar{y}_i)^2}\sqrt{\sum_i^N (p_i - \bar{p}_i)^2}}$$

- Symmetric mean absolute percentage error. This is an alternative to MAPE (mean absolute percentage error) which tends to blow up to infinity when the true data has zero values:

$$\text{sMAPE} = \frac{1}{N}\sum_i^N \frac{|y_i - p_i|}{|y_i| + |p_i|}$$

- Critical Success Index

$$\text{CSI} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

where TP are true positives (cells with both the predictions and ground truths greater than γ), FP are false positives (cells whose ground truths are less than γ but the model's predictions are greater than γ), and FN are false negatives (cells where the model fail to predict a flooded area). In our experiments, we consider $\gamma = 0.001$ m which is the lowest positive value possible.

5.3. Results

We compare our method against the following baseline methods.

- **GCN-GRU**: This model is similar to FloodGNN-GRU, but we simply use a graph convolution network (GCN) [36] that processes velocities and scalar features by concatenating them with other features. This method will also serve as part of our ablation analysis evaluating the impact of vector feature representations.
- **MLP-GRU**: This model is also similar to FloogGNN-GRU, but we use a one-layer multi-layer perceptron (MLP) instead of a GNN layer. As a consequence, this model does not consider the spatial model.
- **Unet** : This model is based on the U-Net architecture proposed in [62]. To explicitly account for the temporal behavior of a system.

These baselines are representative methods from the literature on machine learning for flood prediction, which include CNN-based methods [16, 19, 18] and RNN-based methods [63, 64].

FloodGNN-GRU was trained using the Adam optimizer [65] with weight decay and learning rate set to 1e-3. The model was trained for 1,000 epochs during which we kept the best-performing state (weights) of our model based on the lowest RMSE score on the validation set. We performed three independent experiments based on three random splits of the data (as described in Section 5.1). For a fair comparison, the baseline methods were trained and tested on the same data splits as our method.

Table 2 shows the means and standard deviations (over 3 independent experiments) of the metric scores for predictions with lead times of 1, 5, and 10 hours. All the methods were trained and validated (using the validation set) for a total of 8 time steps, but the inference is not restricted to a fixed number of time steps. We found that training on fewer time steps than 8 did not perform well at inference time, and we did not notice much improvement when training using a number of time steps greater than 8. The results show that FloodGNN-GRU achieves the best results across different metrics. Based on the results at time $t = 10$, there is a gain of about 17% in terms of RMSE, and 15% in terms of sMAPE, showing that FloodGNN-GRU's prediction of water depth values are the closest to the true water depths. We can also notice that FloodGNN-GRU has a very high NSE score (about 77% increase), proving that FloodGNN-GRU isn't just predicting the trivial mean value of water depth values. The 31% increase in

Pearson's coefficient of correlation score indicates that FloodGNN-GRU's predictions follow the trend of the ground truth better than competing baselines.

Table 2: Predictions time-step of size 1-hour. For each metric, the mean and standard deviation over 3 random, independent experiments are provided. RMSE is in meters; while the rest of the metrics are unitless. ↓ means lower is better, and ↑ means higher is better.

time-step t	RMSE ↓	NSE ↑	r ↑	sMAPE ↓	CSI ↑
Prediction at 1-hour lead-time					
GCN-GRU	.0079 ± .00	.0171 ± .40	.6618 ± .06	.2069 ± .03	.4316 ± .20
MLP-GRU	.0074 ± .00	.1767 ± .15	.6007 ± .06	.2416 ± .02	.7109 ± .02
Unet	.0071 ± .00	.2427 ± .11	.5902 ± .05	.2374 ± .04	.6630 ± .03
FloodGNN-GRU	.0040 ± .00	.7651 ± .02	.8840 ± .01	.1535 ± .01	.8088 ± .02
Prediction at 5-hour lead-time					
GCN-GRU	.0215 ± .00	.1911 ± .05	.4853 ± .04	.2677 ± .03	.6607 ± .02
MLP-GRU	.0219 ± .00	.1592 ± .01	.4422 ± .01	.2631 ± 0.02	.6589 ± .02
Unet	.0218 ± .00	.1698 ± .03	.4427 ± .05	.2729 ± .02	.5949 ± .05
FloodGNN-GRU	.0174 ± .00	.4724 ± .01	.6978 ± .01	.2082 ± .00	.7465 ± .01
Prediction at 10-hour lead-time					
GCN-GRU	.0355 ± .00	.0884 ± .02	.4331 ± .04	.3034 ± .02	.6391 ± .01
MLP-GRU	.0355 ± .00	.0875 ± .02	.3819 ± .01	.2935 ± .01	.6180 ± .03
Unet	.0357 ± .00	.0774 ± .02	.3987 ± .04	.3107 ± .02	.5805 ± .04
FloodGNN-GRU	.0293 ± .00	.3790 ± .02	.6301 ± .01	.2492 ± .00	.6970 ± .03

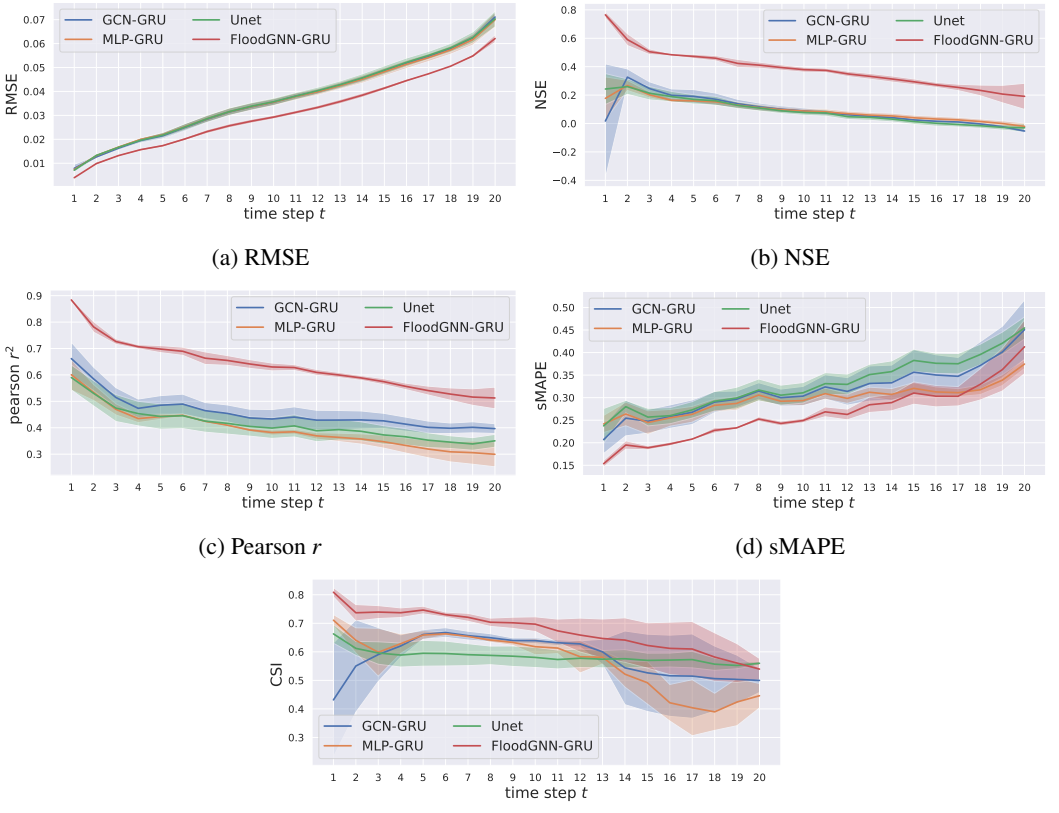
The results of the velocity predictions are shown in Table 3. Here again, we can see that FloodGNN-GRU performs the best, with an RMSE of one order of magnitude less than the second-best approach. This can be attributed to the fact FloodGNN-GRU treats velocities as physical entities by representing them as vector features instead of scalar features.

Table 3: RSME on the velocity predictions in terms of mean and standard deviation values over 3 random, independent experiments.

time	GCN-GRU	MLP-GRU	Unet	FloodGNN-GRU
$t = 1$	.01281 ± .003	.00713 ± .000	.01539 ± .002	.00087 ± .000
$t = 5$	.01407 ± .002	.00607 ± .000	.02505 ± .007	.00085 ± .000
$t = 10$	.01523 ± .003	.00601 ± .000	.02552 ± .007	.00071 ± .000

Figure 5 shows the performances of all the methods over longer lead times (from $t = 1$ to $t = 20$). As expected, the error of the predictions for all methods increases significantly with the lead time. This is due to the accumulation of prediction errors by the recursive model over time. Still, we can observe that FloodGNN-GRU achieves the best results among the approaches considered. As a future work, we will investigate how to incorporate some of the physics of flooding—i.e. the fluid mechanics equations—into our model as a means to improve its long-term accuracy.

Figures 6, 7, and 8 provide visualizations of the correlation between true and predicted values—with respective Pearson's correlation coefficient—at time $t = 1$, $t = 5$, and $t = 10$, respectively, for FloodGNN-GRU and the baselines. The results confirm that FloodGNN-GRU produces the predictions that are the most aligned with the ground truth. However, as noticed earlier, the degradation of the performance is noticeable as the lead time increases. These results also illustrate how all the ML models



(a) RMSE

(b) NSE

(c) Pearson r

(d) sMAPE

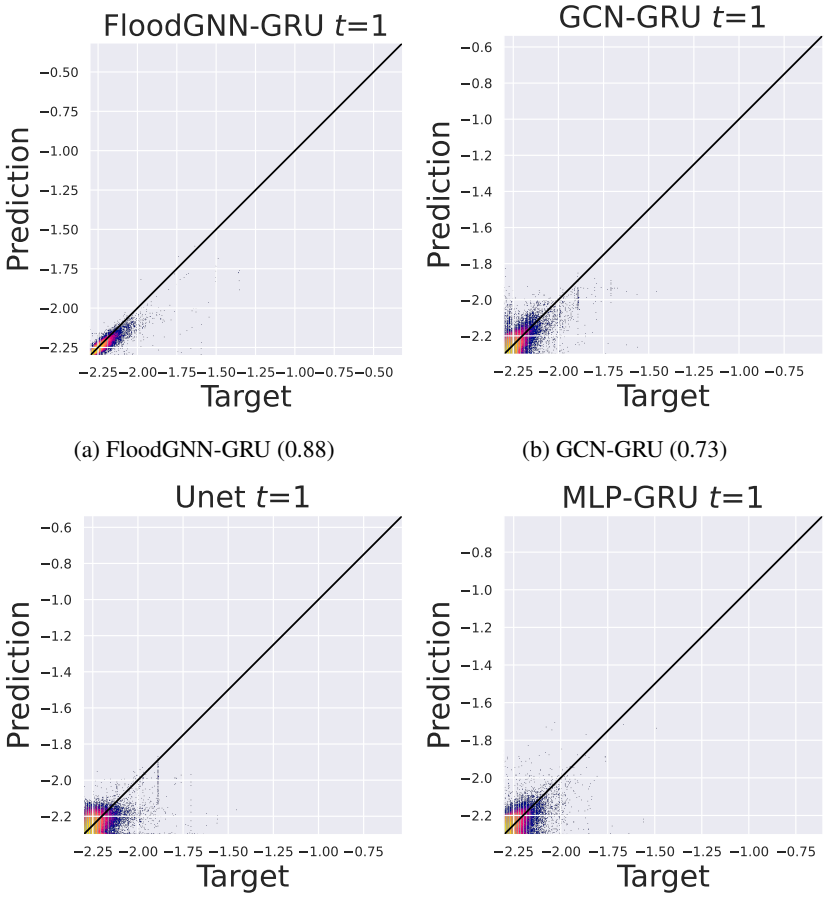
(e) CSI @ 0.001

Figure 5: Predictions from time $t = 1$ to $t = 20$ (in hours) with 1-hour time intervals. Each solid line represents the mean over three experiments, and the shadows along the solid lines represent standard deviations. The results show that FloodGNN-GRU achieves the best results compared to other methods.

tend to underestimate larger values of water depth. This is an indication that ML models for flood prediction still need to be improved for extreme events.

To further assess the accuracy of FloodGNN-GRU, Figure 9 shows visualizations of the true water depth and water depth values predicted by our model over a sub-region sampled from the test dataset for lead times $t = 5$, $t = 10$, and $t = 20$. We can see that FloodGNN-GRU follows the trend of the flooding event represented by the true water depth. However, some of the water depths are underestimated by our model. In other words, FloodGNN-GRU is better at localizing the flood than at predicting its intensity.

Runtime results: One of the motivations for ML-based flood prediction models is their efficient computation compared to traditional physics-based inundation models. Therefore, we compare the running time of LISFLOOD-FP against the training and test time of FloodGNN-GRU. All the experiments were run on a Linux machine with an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz, 256 GB of RAM, and an NVIDIA GPU Ampere A40. Notice that LISFLOOD was run on the CPU of the same machine. The results are shown in Table 4. FloodGNN-GRU can be trained in 4 min 17 sec per epoch (roughly 1 hour and a half for 1000 epochs) and tested in 30 seconds, which is 1000x faster than the time necessary to run the LISFLOOD-FP simulation. Note that simulations are still needed for training FloodGNN-GRU but are not included in the training time. The most relevant runtime result is for testing, which shows that once trained, FloodGNN-GRU is an efficient alternative to running new LISFLOOD-FP simulations.



(a) FloodGNN-GRU (0.88)

(b) GCN-GRU (0.73)

(c) Unet (0.65)

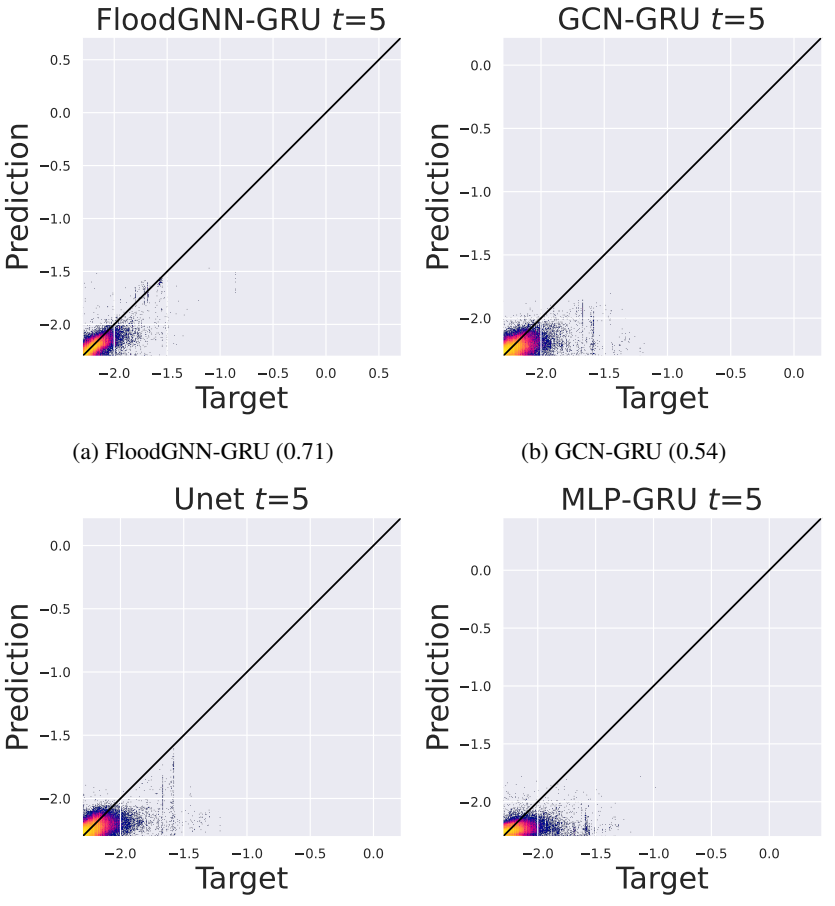
(d) MLP-GRU (0.65)

Figure 6: Scatter plot of water depths in log-log scale at time $t = 1$. FloodGNN-GRU produces predictions that are the most aligned with true values. Pearson's coefficient of correlation values are given at the bottom of each plot.

Table 4: Computation times.

	LISFLOOD-FP	FloodGNN-GRU Training	FloodGNN-GRU Testing
Time (in sec.)	40,780	4,285	30

Ablation analysis. Figure 10 shows the performance of FloodGNN-GRU when: (i) we consider 2 FloodGNN layers instead of 1 and (ii) and when we train it with the L_2 loss instead of L_1. We can see that a single FloodGNN layer performs better than two layers; thus, adding more layers can hurt performance. Furthermore, L_1 allows better learning than L_2 with a significant boost in performance. Figure 11 shows the results when the GRU module is removed from FloodGNN-GRU, we can notice that without the GRU module, the training is not stable and the performance is worsened.



(a) FloodGNN-GRU (0.71)

(b) GCN-GRU (0.54)

(c) Unet (0.51)

(d) MLP-GRU (0.45)

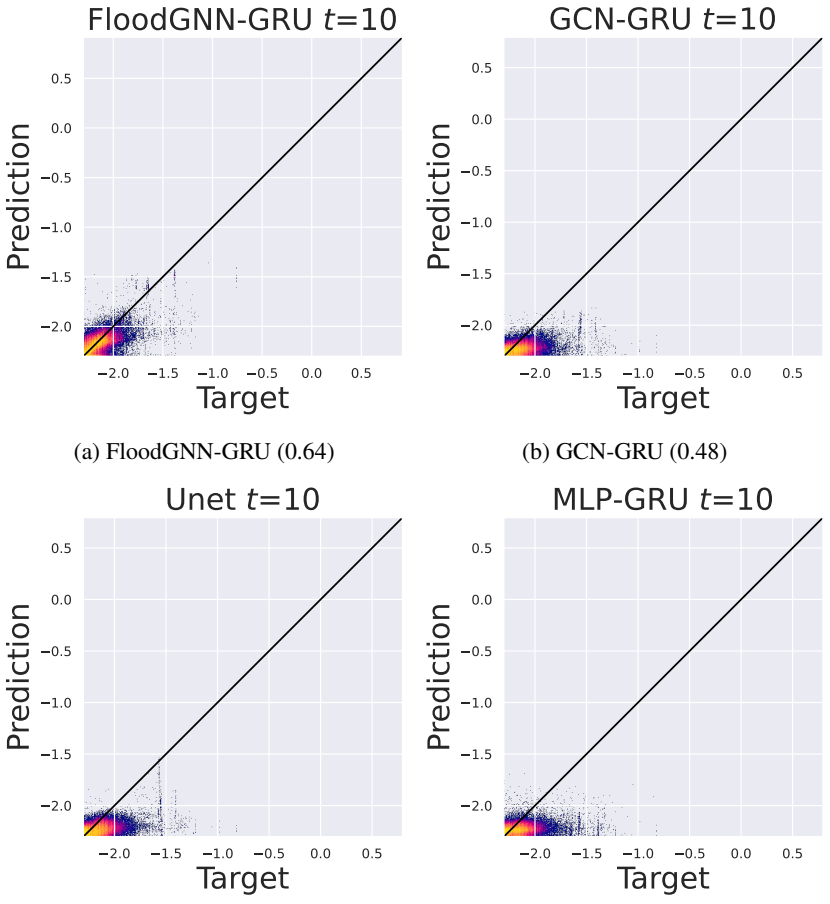
Figure 7: Scatter plot of water depths in log-log scale at time $t = 5$. FloodGNN-GRU produces predictions that are the most aligned with true values. Pearson's coefficient of correlation values are given at the bottom of each plot.

6. Conclusion and Future Work

We have presented FloodGNN-GRU, a graph neural network for flood prediction that captures the dynamics of the flooding event using a GRU architecture. FloodGNN-GRU takes as input a spatially distributed rainfall event, static (e.g. ground elevation, friction), and dynamic (e.g. current water depth and velocity vectors) inputs over a region to predict the next water depth and associated velocities. We propose representing velocities as vector features to preserve their physical properties.

Our experiments were based on a realistic LISFLOOD-FP simulation of Hurricane Harvey in Houston, TX. Results have shown that FloodGNN-GRU outperforms three alternative approaches in terms of different evaluation metrics (RMSE, NSE, r, and sMAP) and for various prediction lead times (from 1-20 hours). Moreover, the training and testing of FloodGNN-GRU require significantly less time than required for running LISFLOOD-FP simulations, about 1000x faster. These results are strong evidence of the potential of data-driven methods to efficiently emulate physics-based inundation models, especially for short-term predictions.

Our work opens several venues for future research. We will investigate how physics can be incorporated into our model—based on the 2-D shallow water equations [1]. We will also leverage watershed



(a) FloodGNN-GRU (0.64)

(b) GCN-GRU (0.48)

(c) Unet (0.45)

(d) MLP-GRU (0.40)

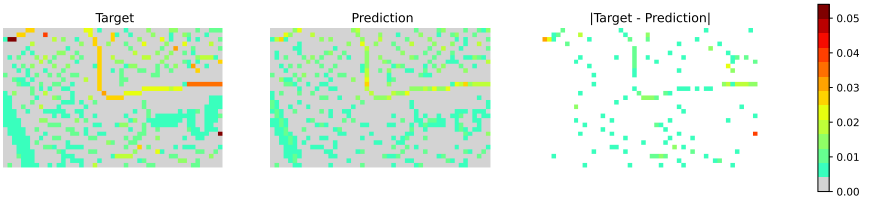
Figure 8: Scatter plot of water depths in log-log scale at time $t = 10$. FloodGNN-GRU produces predictions that are the most aligned with true values. Pearson's coefficient of correlation values are given at the bottom of each plot.

information to better partition regions for training/testing. Finally, our experiments applied a fixed grid (mesh) topology. We will develop adaptive re-meshing schemes able to select the optimal number of nodes/cells for different areas (e.g. urban vs. rural).

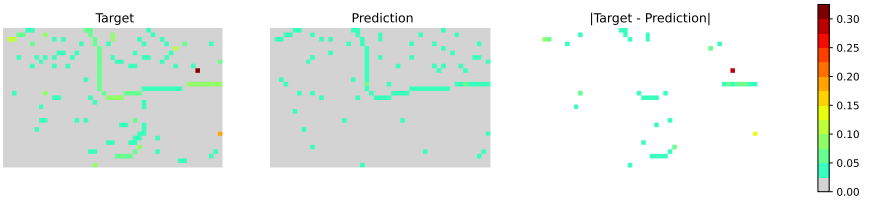
Author contributions: Conceptualization: A.K., J.D.G, A.Se., A.Si.; Data curation: A.K., J.D.G, A.Se., A.Si.; Formal analysis: A.K., A.Si.; Funding acquisition: A.Si.; Investigation: A.K., A.Si.; Methodology: A.K., J.D.G, A.Se., A.Si.; Software: A.K.; Supervision: A.Si.; Validation: A.K.; Visualization: A.K.; Writing—original draft: A.K., A.Si.; Writing—review and editing: A.K., J.D.G., A.Se., A.Si.

Competing interests: The authors declare no competing interests exist.

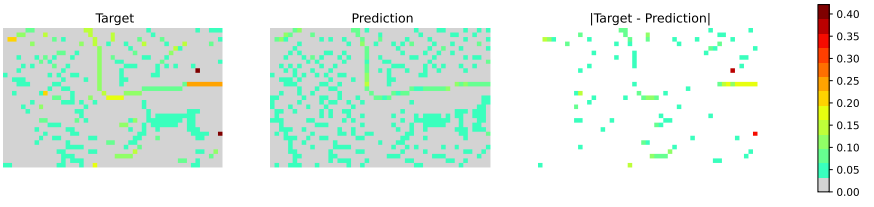
Ethics Statement: The research meets all ethical guidelines, including adherence to the legal requirements of the study country.



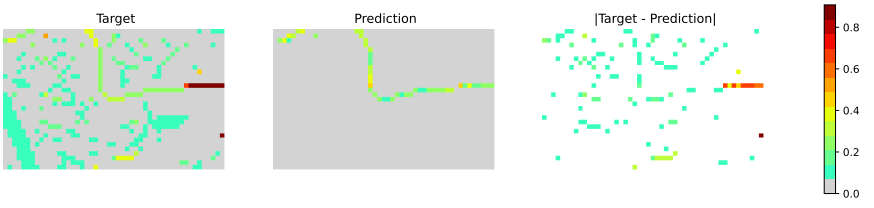
(a) Time $t = 1$



(b) Time $t = 5$

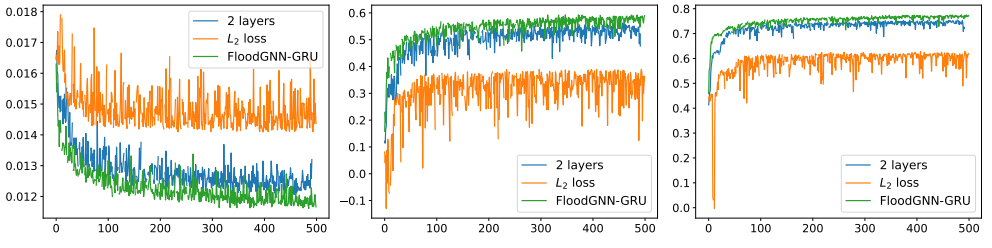


(c) Time $t = 10$



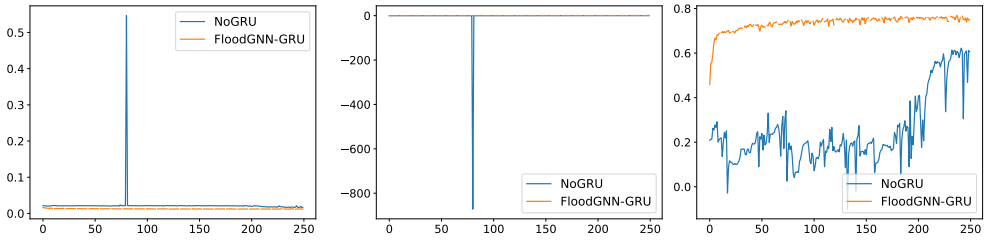
(d) Time $t = 20$

Figure 9: True water depth in meters (Column 1) compared to FloodGNN-GRU predictions (Column 2), and the water depth difference map (Column 3). Our approach can localize the flood (i.e., locations with larger water depths), but the extent of the flood is often underestimated, which is consistent with the results from Figures 6-8. Note that the color scales are different for each time to better appreciate the difference between the target and prediction values at the different water depth levels.



(a) RMSE (b) NSE (c) r

Figure 10: Performance when FloodGNN-GRU is trained with 2 FloodGNN layers instead of 1, and when it is trained with L_2 loss instead of L_1.



(a) RMSE (b) NSE (c) r

Figure 11: Performance when the GRU module of FloodGNN-GRU is removed.

Data Availability Statement: The implementations and datasets used in this work are available at https://zenodo.org/doi/10.5281/zenodo.11416792.. Please README.md file for further instructions

Funding statement: A.K. and A.Si. were supported by a Partners of the Americas US-Brazil 100K Strong Program Artificial Intelligence for Urban Sustainability and Resilience to Natural Disasters in the Americas and by the US Department of Transportation (USDOT) Tier-1 University Transportation Center (UTC) Transportation Cybersecurity Center for Advanced Research and Education (CYBER-CARE) (Grant No. 69A3552348332). A.Se. was supported by the Texas General Land Office (Contract No. 19-181-000-B574).

References

[1] Paul D. Bates. Flood inundation prediction. *Annual Review of Fluid Mechanics*, 54(1):287–315, 2022.

[2] Simon Berkhahn, Lothar Fuchs, and Insa Neuweiler. An ensemble neural network model for real-time prediction of urban floods. *Journal of hydrology*, 575:743–754, 2019.

[3] S. Yu. Schreider, D. I. Smith, and A. J. Jakeman. Climate change impacts on urban flooding. *Climatic Change*, 47(1): 91–115, Oct 2000.

[4] Bates. The lisflood-fp flood inundation model, 2013.

[5] Amir Mosavi, Pinar Ozturk, and Kwok-wing Chau. Flood prediction using machine learning models: literature review. *Water*, 10(11):1536, 2018.

[6] Farzad Piadeh, Kourosh Behzadian, and Amir Alani. A critical review of real-time modelling of flood forecasting in urban drainage systems. *Journal of Hydrology*, page 127476, 2022.

[7] Sean A Woznicki, Jeremy Baynes, Stephanie Panlasigui, Megan Mehaffey, and Anne Neale. Development of a spatially complete floodplain map of the conterminous United States using random forest. *Science of the total environment*, 647: 942–953, 2019.

[8] Elyssa L Collins, Georgina M Sanchez, Adam Terando, Charles C Stillwell, Helena Mitasova, Antonia Sebastian, and Ross K Meentemeyer. Predicting flood damage probability across the conterminous united states. *Environmental Research Letters*, 17(3):034006, 2022.

[9] Qing Lin, Jorge Leandro, Wenrong Wu, Punit Bhola, and Markus Disse. Prediction of maximum flood inundation extents with resilient backpropagation neural network: case study of Kulmbach. *Frontiers in Earth Science*, page 332, 2020.

[10] Sella Nevo, Efrat Morin, Adi Gerzi Rosenthal, Asher Metzger, Chen Barshai, Dana Weitzner, Dafi Voloshin, Frederik Kratzert, Gal Elidan, Gideon Dror, and others. Flood forecasting with machine learning models in an operational framework. *Hydrology and Earth System Sciences Discussions*, pages 1–31, 2021.

[11] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080.

[12] Gang Zhao, Bo Pang, Zongxue Xu, Dingzhi Peng, and Depeng Zuo. Urban flood susceptibility assessment based on convolutional neural networks. *Journal of Hydrology*, 590:125235, 2020. ISSN 0022-1694.

[13] Dieu Tien Bui, Nhat-Duc Hoang, Francisco Martínez-Álvarez, Phuong-Thao Thi Ngo, Pham Viet Hoa, Tien Dat Pham, Pijush Samui, and Romulus Costache. A novel deep learning neural network approach for predicting flash flood susceptibility: A case study at a high frequency tropical storm area. *Science of The Total Environment*, 701:134413, 2020. ISSN 0048-9697.

[14] Yi Wang, Zhice Fang, Haoyuan Hong, and Ling Peng. Flood susceptibility mapping using convolutional neural network frameworks. *Journal of Hydrology*, 582:124482, 2020. ISSN 0022-1694.

[15] Zifeng Guo, Vahid Moosavi, and João P. Leitão. Data-driven rapid flood prediction mapping with catchment generalizability. *Journal of Hydrology*, 609:127726, 2022. ISSN 0022-1694.

[16] Roland Löwe, Julian Böhm, David Getreuer Jensen, Jorge Leandro, and Søren Højmark Rasmussen. U-flood – topographic deep learning for predicting urban pluvial flood water depth. *Journal of Hydrology*, 603:126898, 2021. ISSN 0022-1694.
[17] Julian Hofmann and Holger Schüttrumpf. floodGAN: Using Deep Adversarial Learning to Predict Pluvial Flooding in Real Time. *Water*, 13(16):2255, 2021.
[18] Syed Kabir, Sandhya Patidar, Xilin Xia, Qiuhua Liang, Jeffrey Neal, and Gareth Pender. A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, 590:125481, 2020. ISSN 0022-1694. http://dx.doi.org/https://doi.org/10.1016/j.jhydrol.2020.125481.
[19] Zifeng Guo, Joao P Leitao, Nuno E Simões, and Vahid Moosavi. Data-driven flood emulation: Speeding up urban flood predictions by deep convolutional neural networks. *Journal of Flood Risk Management*, 14(1):e12684, 2021.
[20] Deng-Lin Chang, Sheng-Hsueh Yang, Sheau-Ling Hsieh, Hui-Jung Wang, and Keh-Chia Yeh. Artificial intelligence methodologies applied to prompt pluvial flood estimation and prediction. *Water*, 12(12):3552, 2020.
[21] Li-Chiu Chang, Fi-John Chang, and Yen-Ming Chiang. A two-step-ahead recurrent neural network for stream-flow forecasting. *Hydrological Processes*, 18(1):81–92, 2004.
[22] Qiao-Feng Tan, Xiao-Hui Lei, Xu Wang, Hao Wang, Xin Wen, Yi Ji, and Ai-Qin Kang. An adaptive middle and long-term runoff forecast model using eemd-ann hybrid approach. *Journal of Hydrology*, 567:767–780, 2018. ISSN 0022-1694.
[23] Frederik Kratzert, Daniel Klotz, Mathew Herrnegger, Alden K. Sampson, Sepp Hochreiter, and Grey S. Nearing. Toward improved predictions in ungauged basins: Exploiting the power of machine learning. *Water Resources Research*, 55(12): 11344–11354, 2019.
[24] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences*, 23(12):5089–5110, 2019.
[25] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.
[26] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021.
[27] Philip B Bedient, Wayne Charles Huber, Baxter E Vieux, et al. *Hydrology and floodplain analysis*, volume 816. Prentice Hall Upper Saddle River, NJ, 2008.
[28] Paul D Bates and APJ De Roo. A simple raster-based model for flood inundation simulation. *Journal of hydrology*, 236 (1-2):54–77, 2000.
[29] Pierfranco Costabile, Carmelina Costanzo, and Francesco Macchione. Performances and limitations of the diffusive approximation of the 2-d shallow water equations for flood simulation in urban and rural areas. *Applied Numerical Mathematics*, 116:141–156, 2017.
[30] I-Feng Kao, Jia-Yi Liou, Meng-Hsin Lee, and Fi-John Chang. Fusing stacked autoencoder and long short-term memory for regional multistep-ahead flood inundation forecasts. *Journal of Hydrology*, 598:126371, 2021.
[31] Qing Lin, Jorge Leandro, Stefan Gerber, and Markus Disse. Multistep flood inundation forecasts with resilient backpropagation neural networks: Kulmbach case study. *Water*, 12(12):3568, 2020.
[32] Roland Löwe, Julian Böhm, David Getreuer Jensen, Jorge Leandro, and Søren Højmark Rasmussen. U-FLOOD–Topographic deep learning for predicting urban pluvial flood water depth. *Journal of Hydrology*, 603:126898, 2021.
[33] Roberto Bentivoglio, Elvin Isufi, Sebastian Nicolaas Jonkman, and Riccardo Taormina. Deep learning methods for flood mapping: A review of existing applications and future research directions. *Hydrology and Earth System Sciences Discussions*, pages 1–43, 2021.
[34] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
[35] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
[36] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.
[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2017.
[38] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
[39] Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.
[40] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
[41] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Message passing neural networks. *Machine learning meets quantum physics*, pages 199–214, 2020.
[42] Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

[43] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[44] Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

[45] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[46] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[47] Kelsey R Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. *arXiv preprint arXiv:2212.03574*, 2022.

[48] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. *arXiv:2002.09405 [physics, stat]*, 2020.

[49] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

[50] Ryan Keisler. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.

[51] Arnold Kazadi, James Doss-Gollin, Antonia Sebastian, and Arlei Silva. Flood prediction with graph neural networks. In *NeurIPS Workshop on Tackling Climate Change with Machine Learning*, 2022.

[52] Roberto Bentivoglio, Elvin Isufi, Sebastiaan Nicolas Jonkman, and Riccardo Taormina. Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. *EGUsphere*, 2023:1–24, 2023.

[53] Paul Bates, Mark Trigg, Jeff Neal, and Amy Dabrowa. Lisflood-fp. *User manual. School of Geographical Sciences, University of Bristol. Bristol, UK*, 2013.

[54] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. http://dx.doi.org/10.3115/v1/W14-4012.

[55] Kyunghyun Cho, Bart Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[56] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[57] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 289–297, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[58] Jiahan Li. Directed weight neural networks for protein structure representation learning, 2022.

[59] Liang Zhang, Guangming Zhu, Lin Mei, Peiyi Shen, Syed Afaq Ali Shah, and Mohammed Bennamoun. Attention in convolutional lstm for gesture recognition. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[60] Steven M Martinaitis, Andrew P Osborne, Micheal J Simpson, Jian Zhang, Kenneth W Howard, Stephen B Cocks, Ami Arthur, Carrie Langston, and Brian T Kaney. A physically based multisensor quantitative precipitation estimation approach for gap-filling radar coverage. *Journal of Hydrometeorology*, 21(7):1485–1511, 2020.

[61] Alfred J. Kalyanapu, Steven J. Burian, and Timothy N. McPherson. Effect of land use-based surface roughness on hydrologic model output. *Journal of Spatial Hydrology*, 9:51–71, 2010. URL https://api.semanticscholar.org/CorpusID:67829520.

[62] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL http://arxiv.org/abs/1505.04597.

[63] Zhice Fang, Yi Wang, Ling Peng, and Haoyuan Hong. Predicting flood susceptibility using lstm neural networks. *Journal of Hydrology*, 594:125734, 2021.

[64] R. Hu, F. Fang, C.C. Pain, and I.M. Navon. Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method. *Journal of Hydrology*, 575:911–920, 2019.

[65] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.