Pluvial Flood Emulation with Hydraulics-informed Message Passing

Arnold Kazadi [1] **James Doss-Gollin** [2] **Arlei Silva** [1]

Abstract

Machine Learning (ML) has emerged as a promising alternative to numerical methods for physics-based simulation due to its flexibility and efficiency. Flood modeling is a key case study for ML-based simulation due to its relevance as a tool for supporting preventive and emergency measures to mitigate flood risks. However, the complexity of the topography or domain (ground elevation) and the sparsity of the time-evolving precipitations (external forcing) can be challenging for most existing ML approaches for simulating flooding processes in space and time. Another critical challenge is incorporating physics domain knowledge (hydraulics) into these data-driven models. This paper addresses these challenges by introducing a hydraulics-informed graph neural network for flood simulation. Given a (geographical) region and precipitation data, our model predicts water depths in an auto-regressive fashion. We propose a message-passing framework inspired by the conservation of momentum and mass expressed in the shallow-water equations, which describe the physical process of a flooding event. Empirical results on a dataset covering 9 regions and 7 historical precipitation events demonstrate that our model outperforms the best baseline, and can capture the propagation of water flow more effectively, especially at the very early stage of the flooding event when the amount of water in the domain is scarce. Differently from some of the most recent methods for ML-based simulation, which tend to work well only when the domain is a smooth surface (e.g., flat terrain), we show that our solution achieves accurate results for real ground elevation data.

[1]Department of Computer Science, Rice University, Houston, TX, USA [2]Department of Civil and Environmental Engineering, Rice University, Houston, TX, USA. Correspondence to: Arnold Kazadi <akn7@rice.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Flooding is the natural hazard with the greatest social and economic impact in the United States and affects lives and livelihoods around the world (Tellman et al., 2021; Eckstein et al., 2021; National Academies of Sciences, Engineering, and Medicine; Jha et al.; PBL Netherlands Environmental Assessment Agency). In cities, flooding generates direct property damage, indirect losses through supply chain disruption, and threats to livelihood through drowning and interruption of transportation networks, limiting access to emergency services (Haraguchi & Lall; Han et al.; Gori et al.; Panakkal et al., a;b). With climate change increasing the intensity and frequency of extreme precipitation in many regions (on Climate Change , IPCC), urbanization reducing natural flood protection (Merz et al., 2014; Sebastian et al., 2019), and rapid population growth in flood-prone regions, the severity of urban flooding is projected to continue to grow (Berkhahn et al., 2019; Schreider et al., 2000).

The evolution of a flooding event is a time-evolving physical process typically represented by the 2D shallow-water equations (Eq. 1 & 2). In practice, flooding is modeled using specialized solvers, such as LISFLOOD-FP (Shaw et al., 2021) and HEC-RAS (Brunner, 2016), which not only solve the relevant PDEs but also account for space- and time-varying rainfall, evolving inundation regions, topographies, and additional features. However, these models remain computationally expensive and require extensive calibration of key parameters such as Manning's roughness coefficients and infiltration rates (Zajac et al., 2013), with documented cases of major calibration failures (Van den Honert & McAneney, 2011). This limits the utility of these specialized solvers for vitally important applications including real-time flood warning, probabilistic hazard assessment, representation of green infrastructure benefits, and optimization of infrastructure design.

Machine learning (ML) methods have emerged as a promising alternative to hydrodynamic numerical models due to their flexibility and efficiency (Mosavi et al., 2018; Bentivoglio et al., 2022) (See running times in Appendix A.6). ML-based models for flooding have generally fallen into three groups. The first uses time series models such as Long-Short Term Memory (LSTM) networks trained on gauge observations to predict the time series of discharge (flow/time) at a location (Wi & Steinschneider, 2022; 2023; Nevo et al., 2021). While this approach has proven flexible and skillful in transfer learning tasks, it fails to capture the spatially varying dynamics. The second approach predicts the maximum extent of a flood, given information on the area affected and the storm, trained on high water marks and satellite observations (Muñoz et al., 2021; Berkhahn et al., 2019; Kabir et al., 2020; Löwe et al., 2021; Hofmann & Schüttrumpf, 2021). However, such approaches do not provide information on the time evolution of the system, critical for many applications, and may be difficult to check for physical realism. The third approach considers both spatial and temporal dynamics of flooding (like the 2D hydrodynamic models). The primary limitation of this approach is that observations are not, in general, available, so models are trained in "surrogate" mode on the output of computationally expensive models (Bates, 2022). This approach is the focus of our paper.

In this work, we propose ComGNN, a hydraulics-inspired graph neural network for flood simulation. GNNs have achieved promising results in predicting physics simulations (Pfaff et al., 2021; Lino et al., 2022), including fluid dynamics problems (Keisler, 2022; Lam et al., 2022). They support a wide range of PDE discretizations, such as regular and irregular meshes (Brandstetter et al., 2022). Our proposed model takes as input a directed graph derived from the flow direction of a region where each mesh cell is a node with an outgoing edge to its steepest neighbor cell. At each time step, each node is first considered as an isolated bucket that accumulates its current water volume and water from the rain, which is later propagated to the surrounding nodes using a message-passing inspired by the conservation of momentum and mass. Extensive experiments show that these features enable our method to simulate flooding events much more accurately than existing approaches, including modern ML models for physics-based simulation. Our work makes the following contributions to the emerging field of GNN-based flood simulation:

- We propose ComGNN, a novel graph neural network for flood modeling given spatially and temporally varying rainfall that operates in a two-stage paradigm: (1) retain water where it falls and (2) propagate water to surrounding areas.
- We propose a message-passing mechanism on the flow direction graph that is explicitly designed based on the conservation of momentum and mass for water propagation. Furthermore, this design handles sparse data better than competitive baselines, being able to predict (shallow) water depths at the beginning of a flooding process when the land is still dry.
- We evaluate our method using 9 watersheds (regions) and 7 historical floods. Our experiments show that our approach outperforms current approaches under multiple settings such as early stage simulation and unseen regions and/or unseen precipitations.

2. Related Work

Machine learning for spatial and temporal variability of floods. The focus in ML for flood prediction has been on modeling either the spatial or temporal variability of floods. For instance, ML has been used to predict water flow over time at a single location (Wi & Steinschneider, 2022; 2023; Nevo et al., 2021). There have been applications of ML to the prediction of flood inundation, susceptibility, and hazard maps (Wang et al., 2020; Guo et al., 2022; Löwe et al., 2021; Oliveira Santos et al., 2023; Farahmand et al., 2023). (Mosavi et al., 2018) provides a comprehensive review of ML approaches for flood prediction. (Bentivoglio et al., 2022) review machine learning applications to flood mappings. However, the interplay between predictions of spatial and temporal variabilities is critical for ML to be used as an alternative to current 2D hydrodynamic models for flood simulation. There have recently been a few works addressing this problem. For instance, (Kazadi et al., 2022; 2024) proposes a GNN for flood prediction in an auto-regressive manner but fails to account for the rainfall. (Bentivoglio et al., 2023) also proposes a GNN architecture inspired by the shallow-water equations. However, it does not account for the rainfall either and considers synthetic topographies based on Perlin noise (Perlin, 2002), which are smoother and simpler than real-world topographies (see Figure 3).

Machine learning for dynamical systems. Machine learning methods for modeling physical processes and solving PDEs can be applied to flood simulation given the similarities in tasks performed. The message passing mechanism of Graph Neural Networks (GNNs) has been associated with approximations to differential operators, suitable for solving PDEs (Brandstetter et al., 2022; Maddix et al., 2022). GNNs have been successfully applied to physics-based simulations (Sanchez-Gonzalez et al., 2019; Kipf et al., 2018; Fortunato et al., 2022; Cranmer et al., 2020; Battaglia et al., 2016; Allen et al., 2022). Neural operators are parametric/learnable approximators of nonlinear operators that learn a mapping from a parameter function and/or the initial condition of a PDE to its solution function. Different architectures of neural operators have been proposed. For instance, based on the universal approximation theorem of operators, DeepONet (Lu et al., 2021) proposes two subnetworks, a branch net and a trunk net, to approximate an operator. More recently, nonlinear operators have been approximated by combining (linear) kernel integral transforms and non-linear activation functions. For instance, GNO (Anandkumar et al., 2019) approximates the integral as a node update using a graph defined on the domain, FNO (Li et al., 2021) regards the integral as a convolution, which is computed as a simple multiplication in the Fourier domain. Spatiotemporal Implicit Neural Representations are space-or/and time-continuous learning models, which, like neural operators, also learn a mapping between functions to solve PDEs (e.g., PINNs (Raissi et al., 2019), DINO (Yin et al., 2023)). These approaches, however, are not scalable as coordinate (in space or time) sampling is intractable on a vast domain. ML for the simulation of dynamical systems has shown promising results in solving physical processes, including weather forecasting (Lam et al., 2022; Keisler, 2022; Pathak et al., 2022). Flood simulation, however, poses new challenges to ML-based simulation due to the nature of real-world datasets, which cover large and complex topographies — as opposed to small and smooth domains considered by existing work. For instance, Sun et al.(2023) applied FNO for flood prediction, but considered a very small domain and only predicted a single lead time. Another challenge with flood simulation is the need to account for precipitation data (external forcing). Our work addresses these challenges by proposing a novel Graph Neural Network for pluvial flood dynamics. Experimental results show that our approach outperforms competitive baselines, including FNO.

3. Flood Modeling: Mathematical Framework

The theoretical framework for flood modeling is based on fluid mechanics described by the 3D Navier-Stokes equation. In practice, however, the characteristic vertical length scale of the flow is very small with respect to the characteristic horizontal length scale, resulting in a constant horizontal velocity field throughout the depth of the fluid. The dynamics of a flooding process are, therefore, derived by depth integrating the 3D Navier-Stokes equation, leading to a system of non-linear PDEs called shallow-water equations (de Almeida et al., 2012), which, without convective acceleration and negligible friction, are defined as follows.

$$\frac{\partial h}{\partial t}+\nabla\cdot\mathbf{q}=0 \quad \text{(conservation of mass)} \tag{1}$$

$$\frac{\partial \mathbf{q}}{\partial t}+gh\nabla(h+z)=\mathbf{0} \quad \text{(conservation of momentum)} \tag{2}$$

where $h(x,y;t)$ is the water depth relative to the ground elevation $z(x,y)$, $\mathbf{q}=(q_x(t),q_y(t))$ is the discharge (per unit width), $\nabla=(\frac{\partial}{\partial x},\frac{\partial}{\partial y})$ is the spatial gradient operator.

4. Problem Formulation and Approach

This section introduces our formulation for the flood simulation problem and describes ComGNN, a GNN for flood simulation based on the retention and dispersion of water.

4.1. Problem Formulation

Given a region $\mathcal{R}$, represented as a graph, and a spatially distributed rainfall event $p^{1:K}$ over K time steps, our goal to predict wader depths $h^{1:K}$ over $\mathcal{R}$ for these K time steps.

4.2. Method

We propose ComGNN, a GNN model that operates in two stages, namely, *water retention* and *water dispersion*.

In the water **retention phase**, we consider each node v_i as an isolated bucket with no water exchange with its adjacent nodes. The water level in v_i is represented by the latent features $\mathbf{e}_i^t(\in\mathbb{R}^d)$, solely depending on the rainfall $p_i^t(\geq 0)$ and previous water level $h_i^{t-1}(\geq 0)$.

$$\mathbf{e}_i^t=\text{MLP}([p_i^t\|h_i^{t-1}]) \tag{3}$$

where MLP is a multi-layer perceptron, and $\|$ is concatenation. The **dispersion phase** acts as a learning-based spatial solver of the shallow-water equations. Following the method of lines (Schiesser, 2012), we first define a scheme for the spatial domain. The n^{th}-derivative of f of order n can be approximated using a Taylor expansion as:

$$\frac{\partial^n f(x)}{\partial x}=\sum_{i=1}^{N}\alpha_i f(y_i) \tag{4}$$

where α_i are coefficients, and y_i are points sampled in the neighborhood of x ($y_i=x+\Delta x_i$). For instance, the first-order forward finite difference approximation can be recovered from Eq 4 by setting $N=2$, $\Delta x_1=0,\alpha_1=-\frac{1}{\Delta x_2}$, and $\alpha_2=\frac{1}{\Delta x_2}$ (See Appendix A.1). When the coefficients α_i are learnable from the sampled points, it gives rise to an adaptive approximation scheme with different orders of accuracy for each point x. Eq 4 can be seen as a special case of the more general message passing ψ and node update operation ϕ in GNNs (Brandstetter et al., 2022).

$$\frac{\partial^n f}{\partial x}(x)=\phi\left(\{\psi(f(y_i),f(x))\}_{y_i\in\mathcal{N}(x)}\right) \tag{5}$$

where $\mathcal{N}(x)$ is the neighborhood of x. By setting ψ to a scaling factor of its first argument $f(y_i)$, and ϕ to the summation of its arguments, we recover Eq 4. Applying the spatial derivative from Eq 5 to the conservation of momentum (Eq. 2) at each node i gives:

$$\frac{\partial \mathbf{q}_i}{\partial t}+g\phi\left(\{\psi(h_j,z_j,h_i,z_i)\}_{v_j\in\mathcal{N}_{\textbf{out}}(i)}\right)=\mathbf{0} \tag{6}$$

where $\mathcal{N}_{\textbf{out}}(i)=\{v_j|v_i\rightarrow v_j\}$. We define the message passing ψ as a backward difference to model the ability of the flow to go from v_i to v_j.

$$\psi(h_j,z_j,h_i,z_i)=\sigma(\mathbf{e}_i)\odot\text{MLP}((\mathbf{e}_i+\mathbf{z}_i)-(\mathbf{e}_j+\mathbf{z}_j)) \tag{7}$$

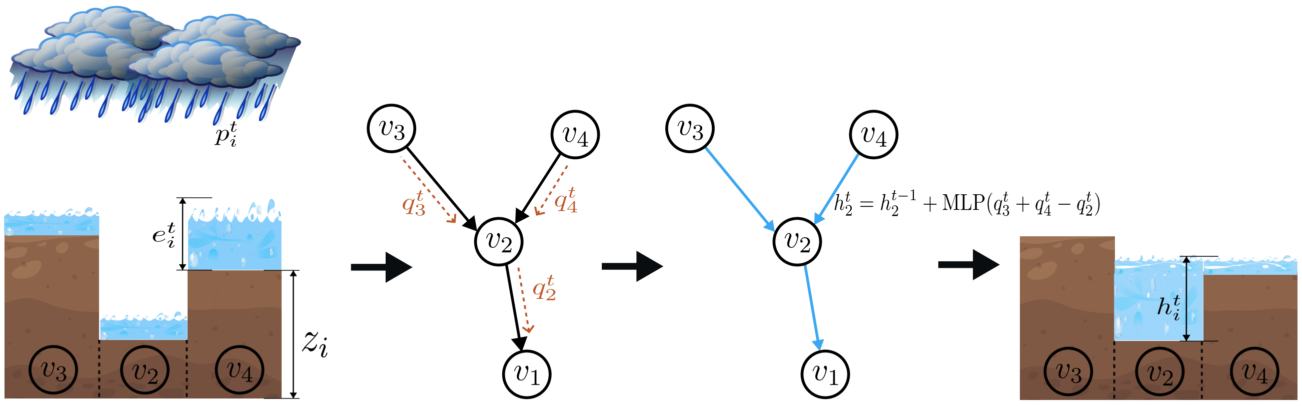


Figure 1: **Water retention and dispersion**. Each cell in the domain representing a region is considered as an isolated bucket filled with water e_i^t from rain p_i^t. Water is propagated by computing, with possibly many iterations, discharges q_i^t (by conserving momentum Eq. 8), and getting the more stable water depth h_i^t (by conserving mass Eq 10).

where σ is the sigmoid function, $\odot$ is the element-wise multiplication, $\mathbf{z}_{i/j} = \text{MLP}(z_{i/j})$, and $\mathbf{e}_{i/j}$ is the latent representation of the water retained from the rain (Eq. 3) which we substitute for the water depth $h_{i/j}$. There is flow from v_i to v_j if there is a difference in water surface, that is, $(\mathbf{e}_i + \mathbf{z}_i) - (\mathbf{e}_j + \mathbf{z}_j)$. This flow can only happen if there is water in v_i, hence, the multiplication by $\sigma(\mathbf{e}_i)$ as a gating mechanism. By defining the node update ϕ as a summation of its arguments and applying the forward Euler time integrator to Eq 6, we have:

$$\mathbf{q}_i^t = \mathbf{q}_i^{t-1} + \Delta t g \sum_{v_j \in \mathcal{N}_{\text{out}}(i)} \sigma(\mathbf{e}_i^t) \odot \text{MLP}((\mathbf{e}_i^t + \mathbf{z}_i) - (\mathbf{e}_j^t + \mathbf{z}_j)) \tag{8}$$

where $\mathbf{q}_i^t$ can be regarded as the total flow going out of v_i.

After obtaining $\mathbf{q}_i^t$ at each cell v_i, the second level message passing of the water dispersion phase is computed with the following (implicit) time integration of the equation capturing the conservation of mass:

$$h_i^t = h_i^{t-1} + \Delta t \phi\left(\{\psi(\mathbf{q}_i^t, \mathbf{q}_j^t)\}_{v_j \in \mathcal{N}_{\text{in}}(i)}\right) \tag{9}$$

where $\mathcal{N}_{\text{in}}(i) = \{v_j | v_j \rightarrow v_i\}$. By setting ψ as an identity function and ϕ as a parametric function (MLP) of the incoming flows from v_j minus outgoing flow of v_i, we obtain:

$$h_i^t = h_i^{t-1} + \Delta t \, \text{MLP}(\sum_{j \in \mathcal{N}_{\text{in}}(i)} \mathbf{q}_j^t - \mathbf{q}_i^t) \tag{10}$$

Multiple iterations of this bi-level message-passing (Eq. 8 & 10) can be performed to simulate dispersion over long distances. Intermediate states can be interpreted as latent space forecasting (Migus et al., 2023) and h_i^t will be the output of the final message-passing—this is analogous to a multistep time integration of h_i^t. At the next time step $t+1$, p_i^{t+1} and h_i^t are fed back into our model, in an auto-regressive manner, for the prediction of h_i^{t+1}. The retention and dispersion processes are illustrated in Figure 1. Because Δt and the gravitational force are fixed—they only appear as constant multiplicative factors—we assume them to be equal to 1 in Eq. 8 & 10.

Region representation as a graph. In practice, region surfaces are represented in a raster format (digital elevation model—DEM), where each pixel/grid cell represents the ground elevation. The first challenge in developing a GNN for flooding simulation is to design a graph topology that captures the dynamics of the flooding process. We convert a given region R into a directed graph $G_R(V, E)$, which remains static. G_R is defined as the D8 flow direction map (Jenson & Domingue, 1988) based on the DEM of R (See Appendix A.2). Each grid cell i is considered as a node v_i, and a single directed outgoing edge $e_{i \rightarrow j} \in E$ connects v_i to its steepest neighbor v_j. Each cell v_i has as features a rainfall time series $p_i^{1:K}$ and ground elevation z_i. Thus, Eq 8 can further be reduced to:

$$\mathbf{q}_i^t = \mathbf{q}_i^{t-1} + \sigma(\mathbf{e}_i^t) \odot \text{MLP}((\mathbf{e}_i^t + \mathbf{z}_i) - (\mathbf{e}_j^t + \mathbf{z}_j))$$

Loss function. At each time t, we propose the following loss function as the objective to minimize the discrepancy between predicted water levels h_i^t and the ground truth w_i^t:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{diff}} + \mathcal{L}_{+} \tag{11}$$

$$\mathcal{L}_{\text{diff}} = \sum_i^N \begin{cases} |h_i^t - w_i^t| & , |h_i^t - w_i^t| < 1 \\ (h_i^t - w_i^t)^2 & , \text{otherwise} \end{cases} \tag{12}$$

$$\mathcal{L}_{+} = \frac{1}{N} \sum_i^N \max(0, -h_i^t) \tag{13}$$

where N is the number of nodes. The loss $\mathcal{L}_{\text{diff}}$ is a combination of the L_1 loss (for very small values) and L_2 (for larger values). $\mathcal{L}_+$ penalizes negative values of h_i^t. In our experiments, $\mathcal{L}_{\text{diff}}$ performs better than L_1 or L_2 used individually. This can be explained by the fact that we are dealing with very sparse data, and the optimization can adapt to different regimes of the learning process.

5. Experiments

We introduce the dataset and evaluation metrics in Section 5.1 and Section 5.2, respectively. We compare our method, called ComGNN, to state-of-art approaches used in ML for simulations (Section 5.3). We present the main results and also investigate how the complexity of the precipitations (sparsity) and the regions (topography) impact the performance of the different methods (Section 5.4).

5.1. Dataset

Experiments are based on the simulations from the hydrodynamic model LISFLOOD-FP (Shaw et al., 2021). We consider 9 sub-watershed regions from Harris County, in Texas (see Figures 4 and 5 and Table 6 in Appendix A.3). For each of these regions, simulations were run using 7 historical rainfall events (based on the flood history in Harris County[1] collected from NOAA NEXRAD radar precipitation records from the Multi-Radar Multi-Sensor Gauge Corrected (MRMS-GC) Quantitative Precipitation Estimation (QPE) product (Martinaitis et al., 2020). See Table 7 in Appendix A.3 for the list of rainfall events.

The Harris County, TX, area, which includes the city of Houston (the 4th most populous in the United States), is the ideal case study for the evaluation of flood simulation methods. The region has experienced multiple severe floods in the past decades and is investing billions of dollars in flood mitigation (HCFCD, 2019) and faces broad climate adaptation challenges representative of those facing urban watersheds across the U.S. (ASFPM, 2020).

There are 63 combinations coming from the 9 sub-watershed regions and 7 rainfall events, of which 9 combinations were used for the training, 3 combinations were used for validation, and the remainder were used for testing.

Data generation The flood data was generated using LISFLOOD-FP (Shaw et al., 2021), a two-dimensional hydrodynamic model specifically designed to simulate floodplain inundation over complex topography by numerically solving the shallow water equations. It predicts water depths in each cell of the discretized domain using an adaptive time stepping. We provided the DEM (ground elevation) of a region and prediction data as input and collected snapshots of water depth states as output every 5 minutes of the simulation process clock-time. It is worth noting that between output intervals, LISFLOOD-FP internally computes several smaller time steps for numerical and computation stability.

Data normalization We normalize the Digital Elevation Model (DEM) of each region independently using standardization. This helps in handling situations where regions have similar topographies but different altitudes as they are expected to show similar inundation behaviors. Since the precipitation and water depth are highly sparse, we log-transform them using $\log(1 + \frac{x}{1e-2})$ (Pathak et al., 2022), followed by a division by 10.

5.2. Evaluation metrics

We apply the root mean square error (RMSE), the Nash–Sutcliffe model efficiency coefficient (NSE), and the Pearson correlation coefficient (r) for accuracy evaluation:

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

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

where y_i is the true value and p_i is the predicted value. We also consider the critical success index (CSI) that measures the spatial accuracy of the classification of cells as flooded or non-flooded areas for a given flooding threshold γ. CSI is evaluated as follows:

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

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 $\gamma = \{0.001\text{ m}, 0.01\text{ m}\}$ since we are dealing with very shallow waters.

5.3. Baselines

We compare our model (ComGNN) to the following approaches (i) **U-net** (Ronneberger et al., 2015), the most popular CNN-based method for the simulation of dynamical systems; (ii) **ConvLSTM** (SHI et al., 2015), an LSTM for weather forecasting on 2D space where the LSTM cell's linear transformations are replaced with CNNs; (iii) **MP-PDE** (Brandstetter et al., 2022), a message passing architecture proposed as a PDE solver; (iv) **FNO** (Li et al., 2021), a neural operator that performs kernel integral in the Fourier space; and (v) **MeshGraphNet** (Pfaff et al., 2021) a GNN-based model for simulating dynamical systems (see Section

[1]https://www.hcfcd.org/About/Harris-Countys-Flooding-History

2 for discussion). We also consider simpler methods such as a multi-layer perceptron (**MLP**) and, more importantly, a method we named **Rain-Incr**, defined as.

$$h_i^t = h_i^{t-1} + \alpha p_i^t$$

where $\alpha \in \mathbb{R}$ is a learnable parameter. Rain-Incr is a one-parameter method based on the intuition that water depth increases proportionally with the amount of rain. Similar to our model (ComGNN), all baselines are used auto-regressively for emulating a flooding event. We do not compare our approach against classical numerical methods such FEM and FVM because the data is generated using LISFLOOD-FP (Shaw et al., 2021), a hydrodynamic model that not only solves the relevant PDEs but also accounts for space- and time-varying rainfall and complex topographies where classical methods can break. More details about implementation are given in Appendix A.4.

Ablated methods To assess the effectiveness of different components of CommGNN, we consider the following ablated variants of our method : (i) **GAT** replaces the water dispersion message passing of CommGNN with a graph attention network (Veličković et al., 2018); (ii) **GCN** is similar to **GAT** but applies a graph convolution network (Kipf & Welling, 2017); (iii) **ComGNN$^-$** applies the precipitation data directly as a node feature, instead of operating in two stages (i.e., retention and then propagation), being similar to the method proposed by (Bentivoglio et al., 2023) but extended to account for rainfall data and using the D8 graph.

Training setup We conducted a thorough hyperparameter search on both our model and baselines and selected the configuration with the lowest RMSE score on the validation set. The simulation lead time was set to 40, the largest we could train on a single NVIDIA GPU Ampere A40. For each sample in the validation set, we trained one model instance, resulting in an ensemble of 3 models per method. (See Appendix A.4 for more details on the methods used)

5.4. Results

Our experiments are based on flood emulations over 40 time steps (the largest that could fit in memory). Table 1 shows results aggregated over the space of a region and over all the regions at time step $t = 20$ and $t = 40$. Our proposed method (ComGNN) achieves the best performance in all the metrics compared to the baselines, with $\sim 20\%$ improvement in RMSE. Surprisingly, Rain-Incr, the intuitive one-parameter method, is competitive with some of the most sophisticated approaches (e.g., MP-PDE, GCN). This demonstrates that a simple method with a strong inductive bias can be a competitive baseline. This observation was also instrumental in the design of ComGNN, as a two-stage mechanism—water retention and dispersion—to better process the external forcing element (i.e., the precipitation).

The effectiveness of our method is demonstrated by the improvement in RMSE of 25 % over ComGNN$^-$ (variant without water retention stage), 40% over GAT, and 35% over GCN, with the last two using different message-passing mechanisms—degree-normalized for GCN and attention-based for GAT—compared with our method (Eq. 8 & 10). Table 1 shows the aggregated results over all the regions and precipitations. In Appendix A.5 we provide a more fine-grained analysis by breaking results from Table 1 down into three categories: (i) unseen regions and unseen precipitations; (ii) unseen regions and seen precipitations; and (iii) seen regions and unseen precipitations. Unseen refers to the data not included in both the training and validation set—notice that training data for a combination of region and precipitation is never leaked to the testing phase.

Precipitation complexity The early stages of a flooding event cover the time from the beginning of the precipitation until when the water starts rising across the domain.

Predicting the early stage is paramount for fully emulating the flooding process, especially for early evacuation warning (Nevo et al., 2021; Piadeh et al., 2022).

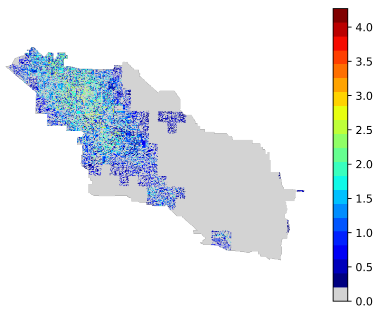
In general, it is more challenging to predict the swift change from dry to wet than the water rise. In Table 2, we reduce the complexity of the problem by using as initial conditions the state of the flood at time $t = 10$ instead of $t = 0$. Here again, we can see that our method outperforms all the baselines and ablated methods. However, it is worth noting that from Table 1 to Table 2 there is a significant improvement for convolution-based methods (ConvLSTM, U-net, FNO) compared to GNN-based ones (GCN, GAT, MeshgGraphNet, MP-PDE, ComGNN$^-$, ComGNN). This can be explained by the graph representation adopted in this work, which is based on the D8 graph. This becomes a limitation under this setting since water can propagate in many directions and there is less dependence on the ground elevation. Learning dynamic graph representations of a region is a direction we want to investigate as future work.

Figure 2 provides a visualization of the absolute errors of our method ComGNN and ConvLSTM based on the results of Table 1. We can see that the predictions made by ComGNN resulted in smaller absolute errors, with a predicted flood extent close to the true flooded area. More similar visualizations are shown in Appendix A.8. We also provide a visualization of the correlations between the predictions and true values as scatter plots in Appendix A.9.

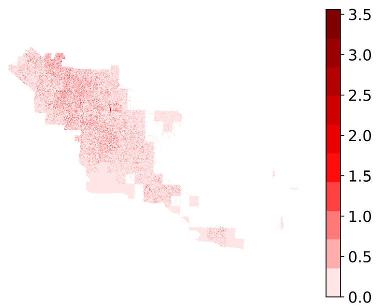
Topography complexity We investigate another dimension of the complexity in flood emulation: ground topography. Our hypothesis is that the complexity of the topography

Table 1: Accuracy/error of simulations over all the test regions and precipitation events combined. The simulation is run over 40 time steps. For CSI, we only show results at $t = 40$ for thresholds $\gamma = \{0.001, 0.01\}$. The results show that our approach achieves the best results across all evaluation metrics.

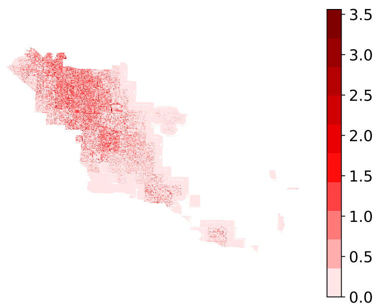
Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r \uparrow$		**CSI** $\uparrow$ $(t = 40)$	
	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$0.001m$	$0.01m$
Rain-Incr	.1972	.5465	.6766	.6590	.7253	.7013	.7122	.4653
ConvLSTM	.1780	.4397	.7199	.7491	.7823	.8181	.7051	.4691
MLP	.2119	.5457	.6445	.6597	.6807	.7156	.5306	.4404
GCN	.1874	.5482	.6986	.6576	.7859	.7669	.7014	.4993
GAT	.2155	.6103	.6366	.6078	.7020	.7042	.6949	.3286
U-net	.2329	.4546	.6001	.7364	.7488	.8022	.6581	.4734
MeshGraphNet	.1597	.4807	.7615	.7141	.8327	.7968	.6120	.5412
MP-PDE	.1824	.5192	.7098	.6817	.7936	.7895	.7158	.5209
FNO	.2162	.5802	.6351	.6317	.6541	.6812	.6582	.3272
ComGNN$^-$	.1571	.4830	.7674	.7121	.8412	.7782	.6180	.5637
ComGNN	**.1328**	**.3615**	**.8218**	**.8154**	**.8866**	**.8854**	**.7463**	**.6486**



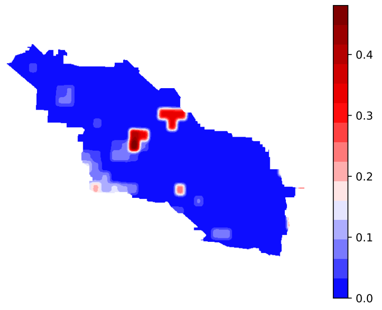
(a) True flooded area at $t = 40$



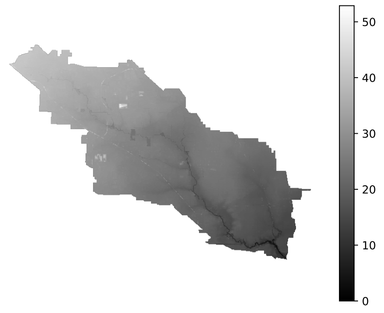
(b) ComGNN error at $t = 40$



(c) ConvLSTM error at $t = 40$



(d) Rainfall at $t = 40$



(e) Ground elevation

Figure 2: Absolute error of our proposed method ComGNN and the absolute error of ConvLSTM compared to the true flood area at lead times 40 (row 1) for the region represented in Figure 2e a precipitation Figure 2d. The results show that ComGNN achieves lower error than the baseline, which is consistent with the results from Table 1.

Table 2: Similar results to those shown in Table1 but using true water depths at time $t = 10$ as initial conditions. We note that at a later stage of the flood—when there is already water in the domain—the baselines perform much better, especially convolution-based ones (ConvLSTM, U-net, FNO). Our approach (ComGNN) still outperforms the baselines in this setting.

Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r \uparrow$		**CSI** $\uparrow$ ($t = 40$)	
	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$0.001m$	$0.01m$
Rain-Incr	.3528	.7205	.6850	.6860	.7409	.7564	.6992	.5978
ConvLSTM	<u>.2749</u>	<u>.5302</u>	<u>.7817</u>	<u>.8014</u>	<u>.8494</u>	<u>.8705</u>	.7081	<u>.6729</u>
MLP	.3871	.7496	.6437	.6687	.6848	.7199	.5516	.5200
GCN	.3461	.7725	.6933	.6552	.7943	.8022	.7086	.5954
GAT	.3882	.8497	.6423	.6110	.7346	.7637	.7117	.5150
U-ne t	.4000	.6431	.6285	.7328	.7462	.8166	.6192	.6096
MeshGraphNet	.3048	.7181	.7444	.6875	.8370	.8142	<u>.7335</u>	.5874
MP-PDE	.3291	.7364	.7142	.6765	.8190	.8099	.7569	.5695
FNO	.3943	.7362	.6352	.6767	.6719	.7470	.6720	.5916
ComGNN^{-}	.2886	.6564	.7647	.7247	.8511	.8131	.6683	.6417
ComGNN	**.2481**	**.5235**	**.8148**	**.8054**	**.8896**	**.8930**	**.7859**	**.7580**

might be one of the reasons why state-of-the-art methods such as MeshgraphNet, MP-PDE, and FNO do not perform well in Table 1. The experiments considered in these works often involve smooth simulation domains, such as 2D planes. In contrast, domains in the real world can present discontinuities—e.g., cliffs (see Figure 3). We, therefore, consider a flood emulation over a flat surface with results at time step $t = 20$ shown in Table 3. Compared with Table 1 and Table 2, we can see that the performance of MeshGraphNet, MP-PDE, and FNO improve with significant margins of 27%, 14%, and 20% in NSE, respectively. These results provide strong evidence that non-smooth surfaces pose a challenge to machine learning-based simulation.

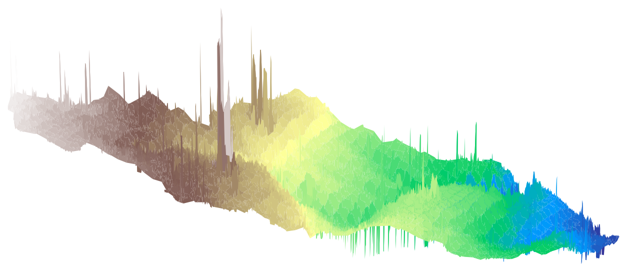


Figure 3: Ground elevation with irregular topography

Table 3: Comparison of the methods on a flat surface. All the methods perform well under these terrain conditions, which are less complex than real-world terrains.

Method	**RMSE** $\downarrow$	**NSE** $\uparrow$	$r \uparrow$
MeshGraphNet	0.3246	0.8697	0.9575
MP-PDE	0.3517	0.8601	0.9673
FNO	0.4619	0.8082	0.9049
ComGNN	**0.2428**	**0.9231**	**0.9868**

Generalization We now evaluate the generalization of the methods to unseen data. Generalization capacity is relevant for flood simulation, as it enables a trained model to be applied to different locations where resources are not available for training a model from scratch. All the methods were trained on one region and precipitation (rainfall event) and then tested on a new region and precipitation data. Table 4 shows the results of this experiment. Based on the NSE metric at time step $t = 20$. The performance of GNN-based methods (MeshGraphnet, MP-PDE, GCN, GAT, and our method ComGNN) is not significantly affected by the new setting, which can be attributed to the ability of GNN architectures to adapt to different inputs (including topologies and features). On the other hand, CNN-based methods (ConvLSTM, U-net, FNO) likely require further data processing tricks, such as data augmentation, to improve generalization. Overall, we note that ComGNN still outperforms the baselines and ablated methods, achieving better generalization.

Conservation of Mass To assess the ability of our method and the baselines regarding conservation of mass in Appendix A.7. We compare the total change of the amount of water (from a time step to the next one) on the ground truth data to the total change of the amount of water in the domain for all models. We show that ComGNN has the lowest deviation from the ground truth.

Table 4: Results on the generalization to new regions and new precipitations. In general, GNN-based methods (GCN, GAT, MeshGraphNet, MP-PDE) generalize better than convolution-based ones (ConvLSTM, U-net, FNO). Our method ComGNN generalizes better than the baselines in terms of most of the evaluation metrics.

Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r \uparrow$		**CSI** $\uparrow$ $(t=40)$
	$t=20$	$t=40$	$t=20$	$t=40$	$t=20$	$t=40$	$0.001m$
Rain-Incr	.1260	1.3642	.6742	.4547	.7601	.7277	<u>.9553</u>
ConvLSTM	.1251	1.2351	.6772	.5043	.7623	**.8038**	**.9584**
MLP	.1320	1.4831	.6533	.4137	.6876	.3018	.7612
GCN	.1017	1.2582	.7606	.4950	.8358	.7598	.9531
GAT	.1241	1.3730	.6807	.4515	.7416	.6447	.9535
U-net	.1481	6.8758	.5995	.0318	.5984	.6363	.9091
MeshGraphNet	<u>.0931</u>	1.1254	.7912	.5506	<u>.8659</u>	.7278	.9196
MP-PDE	.1078	1.2924	.7385	.4816	.8328	<u>.7912</u>	.9525
FNO	.1536	1.3376	.5821	.4645	.5432	.6324	.6784
ComGNN$^-$	.0968	<u>1.0899</u>	<u>.7780</u>	<u>.5664</u>	.8526	.7367	.8755
ComGNN	**.0809**	**1.0538**	**.8337**	**.5829**	**.8968**	.7698	.9513

6. Conclusion

We have presented ComGNN, a hydraulics-informed graph neural network for early-stage flood simulation based on a given rainfall event. ComGNN operates in two stages: at each time step, water from the rain is first stored in the area of the region where it falls (*water retention*), and it is then propagated to the surrounding areas (*water dispersion*) using a message-passing that mimics the conservation of momentum and mass of the shallow-water equations. A region is represented as a directed graph by linking each cell/node to its steepest neighbor based on the D8 flow direction of the region's topography.

Our experiments were based on realistic simulations of 7 historical floods over 9 watershed regions. Results have shown that ComGNN is effective at simulating flooding events in different settings (sparse conditions, new regions, new precipitation data), outperforming existing methods in terms of multiple evaluation metrics (RMSE, NSE, Pearson's coefficient of correlation, and CSI). We also show that current ML methods for dynamical systems and solving PDEs tend to perform the best for smooth surfaces such as 2D flat surfaces, but fail to adapt to the complexity encountered in the topography of real-world surfaces.

In future work, we will improve the graph representation of a region. We will investigate how to dynamically change the graph representation of a region based on the current water surface elevation (water depth + ground elevation) and potential energy surface. The Shallow Water Equations, which are the basis for our model, have multiple applications beyond flooding, including tsunami (Geyer & Quirchmayr, 2018) and atmospheric modeling (Behrens, 1998). The framework of flow graphs (Bressan et al., 2014; Silva et al., 2021; Kocayusufoglu et al., 2022; Smith et al., 2022), which encompasses several scenarios where a physical quantity is transmitted through a graph topology (e.g., traffic, power, water, gas), also provides many potential future generalizations for our work.

Software and Data

The codebase and datasets used in our experiments can be accessed via the repository at `https://github.com/kanz76/ComGNN.git`

Impact Statement

This paper presents a work that aims to advance the field of Machine Learning towards effective and data-driven flood modeling. There are many potential societal consequences of our work, as flooding impacts many communities worldwide. We recognize the importance of accounting for equity considerations in evaluating ML models for flood simulation and highlight the need for further research in this direction before these methods are deployed in practical settings.

Acknowledgements

We gratefully acknowledge the support by a Partners of the Americas US-Brazil 100K Strong Program Artificial Intelligence for Urban Sustainability and Resilience to Natural Disasters in the Americas, 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), and a Rice University Ken Kennedy Institute (K2I) Seed Fund.

References

Allen, K. R., Rubanova, Y., Lopez-Guevara, T., Whitney, W., Sanchez-Gonzalez, A., Battaglia, P., and Pfaff, T. Learning rigid dynamics with face interaction graph networks. *arXiv preprint arXiv:2212.03574*, 2022.

Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., and Stuart, A. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

ASFPM. Urban flood hazards: Challenges and opportunities, 2020. URL https://asfpm-library.s3-us-west-2.amazonaws.com/ASFPM_Pubs/ASFPM_Stormwater_Committee_Urban_Flood_Hazard_Areas_Discussion_2020.pdf.

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

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

Behrens, J. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. *Applied Numerical Mathematics*, 26(1-2):217–226, 1998.

Bentivoglio, R., Isufi, E., Jonkman, S. N., and Taormina, R. Deep learning methods for flood mapping: a review of existing applications and future research directions. *Hydrology and Earth System Sciences*, 26(16):4345–4378, 2022.

Bentivoglio, R., Isufi, E., Jonkman, S. N., and Taormina, R. Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. *EGUsphere*, 2023:1–24, 2023.

Berkhahn, S., Fuchs, L., and Neuweiler, I. An ensemble neural network model for real-time prediction of urban floods. *Journal of hydrology*, 575:743–754, 2019.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

Bressan, A., Čanić, S., Garavello, M., Herty, M., and Piccoli, B. Flows on networks: recent results and perspectives. *EMS Surveys in Mathematical Sciences*, 1(1):47–111, 2014.

Brunner, G. W. Hec-ras river analysis system: Hydraulic reference manual, version 5.0. *US Army Corps of Engineers–Hydrologic Engineering Center*, 547, 2016.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

de Almeida, G. A. M., Bates, P., Freer, J. E., and Souvignet, M. Improving the stability of a simple formulation of the shallow water equations for 2-d flood modeling. *Water Resources Research*, 48(5), 2012.

Eckstein, D., Künzel, V., and Schäfer, L. *The global climate risk index 2021*. Bonn: Germanwatch, 2021.

Farahmand, H., Xu, Y., and Mostafavi, A. A spatial–temporal graph deep learning model for urban flood nowcasting leveraging heterogeneous community features. *Scientific Reports*, 13(1), Apr 2023.

Fortunato, M., Pfaff, T., Wirnsberger, P., Pritzel, A., and Battaglia, P. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

Geyer, A. and Quirchmayr, R. Shallow water equations for equatorial tsunami waves. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2111):20170100, 2018.

Gori, A., Gidaris, I., Elliott, J. R., Padgett, J., Loughran, K., Bedient, P., Panakkal, P., and Juan, A. Accessibility and recovery assessment of Houston's roadway network due to fluvial flooding during Hurricane Harvey. 21(2):04020005. doi: 10.1061/(asce)nh.1527-6996.0000355. URL http://ascelibrary.org/doi/full/10.1061/%28ASCE%29NH.1527-6996.0000355.

Guo, Z., Moosavi, V., and Leitão, J. P. Data-driven rapid flood prediction mapping with catchment generalizability. *Journal of Hydrology*, 609:127726, 2022. ISSN 0022-1694.

Han, I., Whitworth, K. W., Christensen, B., Afshar, M., An Han, H., Rammah, A., Oluwadairo, T., and Symanski, E. Heavy metal pollution of soils and risk assessment in Houston, Texas following Hurricane Harvey. 296:118717.

Haraguchi, M. and Lall, U. Flood risks and impacts: A case study of Thailand's floods in 2011 and research questions for supply chain decision making. 14:256–272. doi: 10.1016/j.ijdrr.2014.09.005. URL http://linkinghub.elsevier.com/retrieve/pii/S2212420914000752.

HCFCD. Prioritization framework for the implementation of the Harris County Flood Control District 2018 bond projects, 2019. URL https://www.harriscountytx.gov/Portals/49/Documents/

Metrics-for-Evaluation-Criteria_2019-August21.pdf.

Hofmann, J. and Schüttrumpf, H. floodGAN: Using Deep Adversarial Learning to Predict Pluvial Flooding in Real Time. *Water*, 13(16):2255, 2021.

Jenson, S. K. and Domingue, J. O. Extracting topographic structure from digital elevation data for geographic information-system analysis. *Photogrammetric Engineering and Remote Sensing*, 54(11):1593–1600, 1988.

Jha, A. K., Bloch, R., and Lamond, J. Cities and flooding: A guide to integrated urban flood risk management for the 21st century. URL https://openknowledge.worldbank.org/handle/10986/2241.

Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., and Pender, G. A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, 590:125481, 2020.

Kazadi, A., Doss-Gollin, J., Sebastian, A., and Silva, A. Flood prediction with graph neural networks. In *NeurIPS Workshop on Tackling Climate Change with Machine Learning*, 2022.

Kazadi, A., Doss-Gollin, J., Sebastian, A., and Silva, A. FloodGNN-GRU: A spatio-temporal graph neural network for flood prediction. (to appear), 2024.

Keisler, R. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *International conference on machine learning*, pp. 2688–2697. PMLR, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kocayusufoglu, F., Silva, A., and Singh, A. K. Flowgen: A generative model for flow graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 813–823, 2022.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations, 2021.

Lino, M., Fotiadis, S., Bharath, A. A., and Cantwell, C. D. Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks. In *AI4Earth Workshop at ICLR*, 2022.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839.

Löwe, R., Böhm, J., Jensen, D. G., Leandro, J., and Rasmussen, S. H. U-FLOOD–Topographic deep learning for predicting urban pluvial flood water depth. *Journal of Hydrology*, 603:126898, 2021.

Maddix, D. C., Saad, N., and Wang, Y. Modeling advection on directed graphs using matérn gaussian processes for traffic flow. In *NeurIPS Workshop on Tackling Climate Change with Machine Learning*, 2022.

Martinaitis, S. M., Osborne, A. P., Simpson, M. J., Zhang, J., Howard, K. W., Cocks, S. B., Arthur, A., Langston, C., and Kaney, B. T. A physically based multisensor quantitative precipitation estimation approach for gap-filling radar coverage. *Journal of Hydrometeorology*, 21 (7):1485–1511, 2020.

Merz, B., Aerts, J., Arnbjerg-Nielsen, K., Baldi, M., Becker, A., Bichet, A., Blöschl, G., Bouwer, L. M., Brauer, A., Cioffi, F., et al. Floods and climate: emerging perspectives for flood risk assessment and management. *Natural Hazards and Earth System Sciences*, 14(7):1921–1942, 2014.

Migus, L., Salomon, J., and patrick gallinari. Stability of implicit neural networks for long-term forecasting in dynamical systems. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

Mosavi, A., Ozturk, P., and Chau, K.-w. Flood prediction using machine learning models: literature review. *Water*, 10(11):1536, 2018.

Muñoz, D. F., Muñoz, P., Moftakhari, H., and Moradkhani, H. From local to regional compound flood mapping with deep learning and data fusion techniques. *Science of the Total Environment*, 782:146927, 2021.

National Academies of Sciences, Engineering, and Medicine. *Framing the Challenge of Urban Flooding in the United States*. National Academies Press. ISBN 978-0-309-48961-4.

Nevo, S., Morin, E., Gerzi Rosenthal, A., Metzger, A., Barshai, C., Weitzner, D., Voloshin, D., Kratzert, F., Elidan, G., Dror, G., and others. Flood forecasting with machine learning models in an operational framework. *Hydrology and Earth System Sciences Discussions*, pp. 1–31, 2021.

Oliveira Santos, V., Costa Rocha, P. A., Scott, J., Thé, J. V. G., and Gharabaghi, B. A new graph-based deep learning model to predict flooding with validation on a case study on the humber river. *Water*, 15(10), 2023.

on Climate Change (IPCC), I. P. *Weather and Climate Extreme Events in a Changing Climate*. Cambridge University Press, 2023.

Panakkal, P., Juan, A., Garcia, M., Padgett, J. E., and Bedient, P. Towards Enhanced Response: Integration of a Flood Alert System with Road Infrastructure Performance Models. pp. 294–305, a.

Panakkal, P., Wyderka, A. M., Padgett, J. E., and Bedient, P. B. Safer this way: Identifying flooded roads for facilitating mobility during floods. b.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., and Anandkumar, A. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators, 2022.

PBL Netherlands Environmental Assessment Agency. The geography of future water challenges. URL https://www.pbl.nl/en/publications/the-geography-of-future-water-challenges.

Perlin, K. Improving noise. *ACM Trans. Graph.*, 21(3): 681–682, jul 2002. ISSN 0730-0301.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Piadeh, F., Behzadian, K., and Alani, A. A critical review of real-time modelling of flood forecasting in urban drainage systems. *Journal of Hydrology*, pp. 127476, 2022.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

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

Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., and Battaglia, P. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Schiesser, W. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Elsevier Science, 2012. ISBN 9780128015513.

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

Sebastian, A., Gori, A., Blessing, R. B., Van Der Wiel, K., and Bass, B. Disentangling the impacts of human and environmental change on catchment response during hurricane harvey. *Environmental Research Letters*, 14 (12):124023, 2019.

Shaw, J., Kesserwani, G., Neal, J., Bates, P., and Sharifian, M. K. LISFLOOD-FP 8.0: the new discontinuous galerkin shallow-water solver for multi-core CPUs and GPUs. 14(6):3577–3602, 2021.

SHI, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and WOO, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Silva, A., Kocayusufoglu, F., Bullo, F., Swami, A., and Singh, A. Combining physics and machine learning for network flow estimation. In *International Conference on Learning Representations*, 2021.

Smith, K. D., Seccamonte, F., Swami, A., and Bullo, F. Physics-informed implicit representations of equilibrium network flows. *Advances in Neural Information Processing Systems*, 35:7211–7221, 2022.

Sun, A. Y., Li, Z., Lee, W., Huang, Q., Scanlon, B. R., and Dawson, C. Rapid flood inundation forecast using fourier neural operator. In *International Conference on Computer Vision (ICCV) Workshops*. IEEE, 2023.

Tellman, B., Sullivan, J. A., Kuhn, C., Kettner, A. J., Doyle, C. S., Brakenridge, G. R., Erickson, T. A., and Slayback, D. A. Satellite imaging reveals increased proportion of population exposed to floods. *Nature*, 596(7870):80–86, Aug 2021. ISSN 1476-4687.

Van den Honert, R. C. and McAneney, J. The 2011 brisbane floods: Causes, impacts and implications. *Water*, 3(4):1149–1173, 2011. ISSN 2073-4441. doi: 10.3390/w3041149. URL https://www.mdpi.com/2073-4441/3/4/1149.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. 2018.

Wang, Y., Fang, Z., Hong, H., and Peng, L. Flood susceptibility mapping using convolutional neural network frameworks. *Journal of Hydrology*, 582:124482, 2020. ISSN 0022-1694.

Wi, S. and Steinschneider, S. Assessing the physical realism of deep learning hydrologic model projections under climate change. *Water Resources Research*, 58(9): e2022WR032123, 2022.

Wi, S. and Steinschneider, S. On the need for physical constraints in deep learning rainfall-runoff projections under climate change. *EGUsphere*, 2023:1–46, 2023.

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., and Gallinari, P. Continuous pde dynamics forecasting with implicit neural representations. In *International Conference on Learning Representations*, 2023.

Zajac, Z., Zambrano-Bigiarini, M., Salamon, P., Burek, P., Gentile, A., and Bianchi, A. Calibration of the lisflood hydrological model for europe. *Calibration Round 2013JRC Technical Report, European Commission, Joint Research Centre, Ispra, Italy*, 2013.

A. Appendix

A.1. Derivative approximation with Taylor Series

For simplicity, let us assume a one-dimensional domain. Suppose we want to approximate the derivate of a smooth enough function f at point x_i with l points to the left, and r points to the right, forming a stencil that includes the points x_j such that $i - l \leq j \leq i + r$. Let us further assume x_j are uniformly, that is, $\Delta x_j = j\Delta x$. The Taylor expansion of f at x_j centered at x_i is

$$f(x_j) = f(x_i) + \frac{j\Delta x}{1!} f_x(x_i) + \frac{(j\Delta x)^2}{2!} f_{xx}(x_i) + \frac{(j\Delta x)^3}{3!} f_{xxx}(x_i) + \frac{(j\Delta x)^4}{4!} f_{xxxx}(x_i) + \ldots$$

where $i - l \leq j \leq i + r$. Multiplying each of these expansions by a constant c_j and summing them up gives

$$\begin{aligned}\sum_{j=i-l}^{i+r} c_j f(x_j) - \left(\sum_{j=i-l}^{i+r} c_j\right) f(x_i) = &\left(\sum_{j=i-l}^{i+r} j c_j\right) \frac{\Delta x}{1!} f_x(x_i) + \left(\sum_{j=i-l}^{i+r} j^2 c_j\right) \frac{(\Delta x)^2}{2!} f_{xx}(x_i) \\ &+ \left(\sum_{j=i-l}^{i+r} j^3 c_j\right) \frac{(\Delta x)^3}{3!} f_{xxx}(x_i) \\ &+ \left(\sum_{j=i-l}^{i+r} j^4 c_j\right) \frac{(\Delta x)^4}{4!} f_{xxxx}(x_i) \\ &+ \ldots \end{aligned} \tag{14}$$

Eq. 14 provides a way to approximate higher order derivatives at any order accuracy of f. For instance, first-order derivative at third-order accuracy can be obtained by setting $\left(\sum_{j=i-l}^{i+r} j c_j\right)$ to 1 and $\left(\sum_{j=i-l}^{i+r} j^2 c_j\right)$ to 0.

A.2. D8 Flow direction graph

We used the tool ArcGIS Pro [2] to generate the D8 flow direction graph of a region based on its digital elevation model (DEM). D8 (eight-direction) indicates that the output direction of a cell is related to its 8 adjacent cells. The direction is coded as an unsigned 8-bit integer, with 1 denoting east, 2 south-east, 4 south, 8 south-west, 16 west, 32 north-west, 64 north, and 128 north-east. We generate a directed graph by considering a cell as a node, and by adding an outgoing edge to the adjacent cell corresponding to the direction code.

Comparison between Flow Direction Graph and Grid-based Graph Since our proposed model ComGNN is based on the flow direction, we instead choose GCN to compare performances when a region is represented as a flow direction and a grid-based graph (when the mesh is directly used as the graph). In Table 5, the GCN with flow direction is denoted as plain GCN, and the one with grid-based graph GCN-grid. We can that with flow direction graph, results are relatively good compared to the grid-based graph.

Table 5: Comparison between D8 flow direction graph and grid-based graph representation.

Method	**RMSE** ↓		**NSE** ↑		r ↑		**CSI** ↑ ($t = 40$)	
	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$0.001m$	$0.01m$
GCN	**.1874**	**.5482**	**.6986**	**.6576**	**.7859**	**.7669**	.7014	**.4993**
GCN-grid	.2046	.5702	.6603	.6397	.7179	.7282	**.7081**	.4239

[2] https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/how-flow-direction-works.htm

A.3. Dataset

Here we provide figures and tables (with descriptions) of the watershed considered in our work.

We consider 9 sub-watershed regions from Harris County in Texas, all shown together in Figure 4.

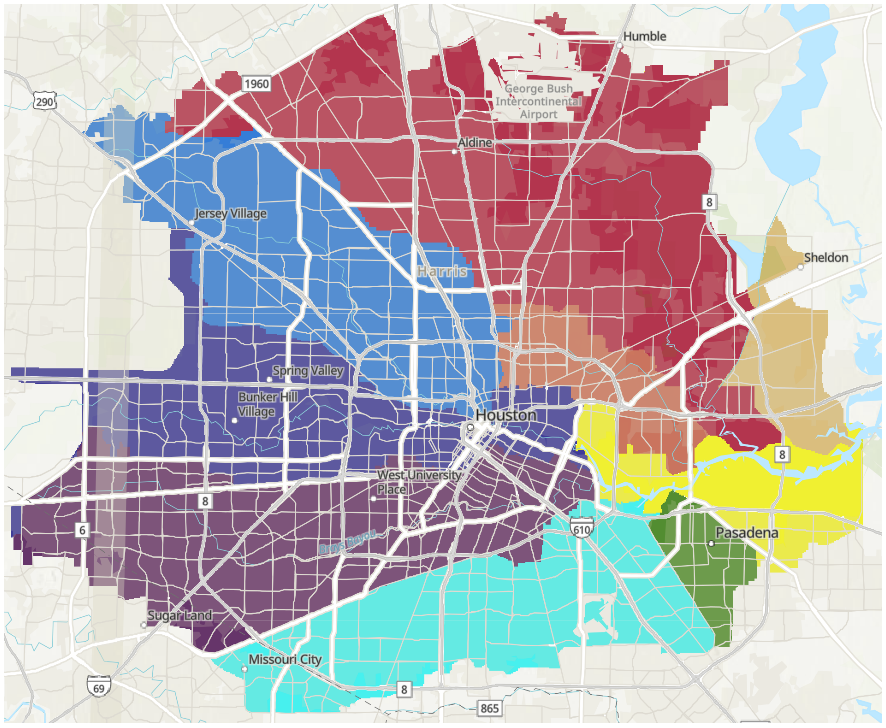


Figure 4: Color-coded watershed regions considered in our work

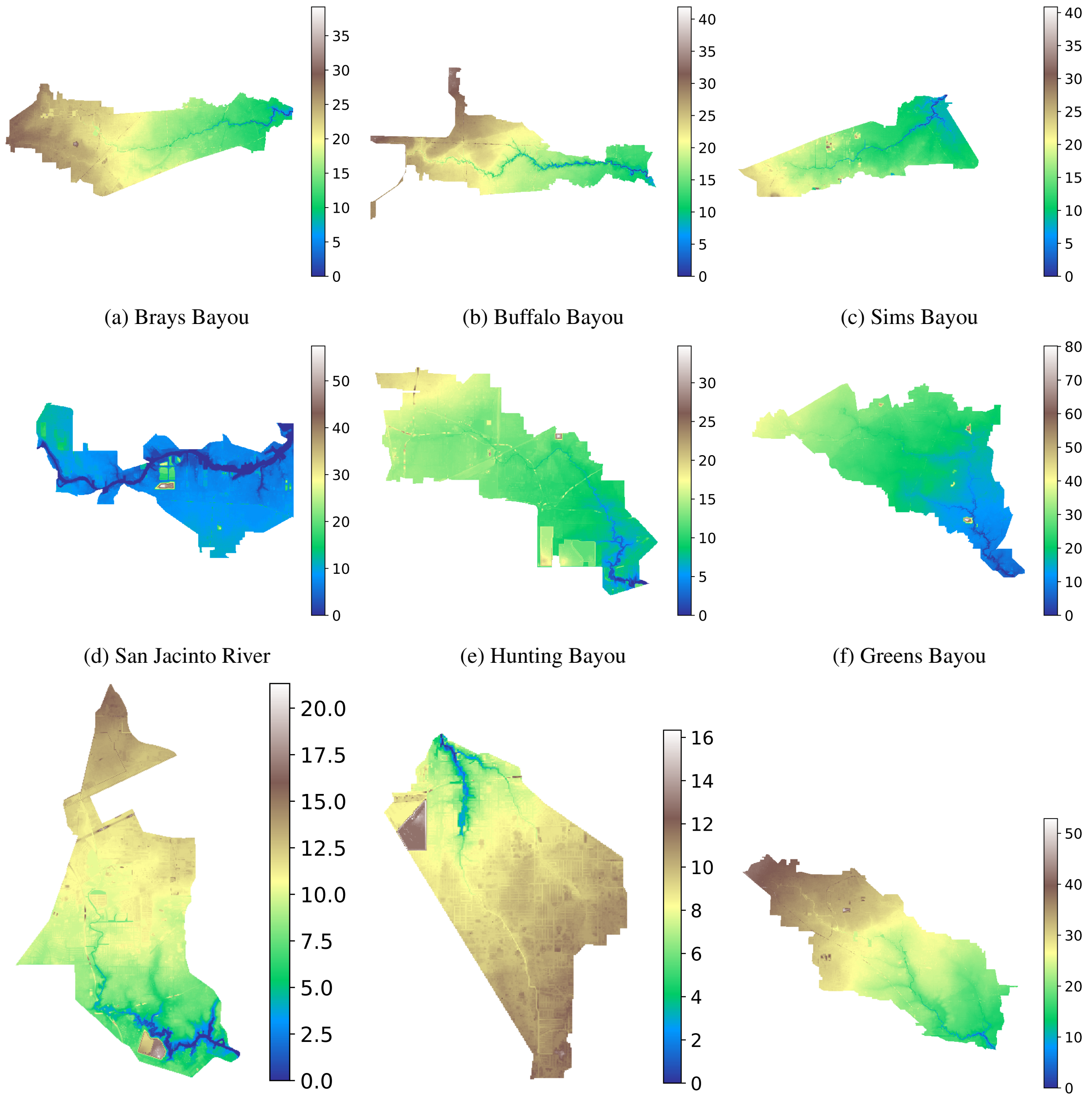
Figure 5 shows the ground elevations of the 9 watersheds. They are represented in a raster format where each pixel cell represents a $30m \times 30m$ area. The areas and dimensions (in terms of number of rows and columns) of the watersheds are given in Table 6. Details about the precipitation data are shown in Table 7.

Table 6: Regions considered in this work with the areas and dimensions in terms of the number of rows and columns in their raster

Region	**Area** (km^2)	**Rows**	**Columns**
White Oak Bayou	288	1083	749
Vince Bayou	41	280	370
Sims Bayou	242	1412	562
San Jacinto River	272	745	406
Hunting Bayou	77	514	417
Greens Bayou	549	1512	1032
Carpenters Bayou	65	331	558
Buffalo Bayou	267	1360	750
Brays Bayou	330	1358	577

A.4. Model Configurations

The learning rate was set to 1e-4 for all the models. We also noticed that all the models considered in our work performed much better with the loss function proposed in Eq. 11, with a bump in performance of up to 30% in some cases. *tanh*



(a) Brays Bayou (b) Buffalo Bayou (c) Sims Bayou

(d) San Jacinto River (e) Hunting Bayou (f) Greens Bayou

(g) Carpenters Bayou (h) Vince Bayou (i) White Oak Bayou

Figure 5: Ground elevations of the 9 watersheds considered in our work

Table 7: Preciptiation data

Rainfall Event	**Date**	**Intensity** (mm/s)	
		mean	max
Pre-Memorial Day Flood	May 13, 2015	0.4	33.9
Memorial Day Flood	May 25, 2015	2.6	97.7
N/A	Oct 31, 2015	6.1	146.5
Tax Day Flood	Apr 17, 2016	3.1	73.3
Hurricane Harvey	Aug 25, 2017	5.6	122.4
N/A	Jul 04, 2018	3.4	85.4
Tropical Storm Imelda	Sep 17, 2019	2.7	103.3

function showed better performance compared to the original activation functions of some of the baselines. All linear transformations were used without the bias term, this seemed to help to deal with the sparsity of the data. The configurations with the best performance of individual methods are given below,

- **Rain-Incr**: simple method we implement as $hi^t = h^{t-1} + \alpha p^t$; where h^t is the water depth at time t, p^t is the precipitation at time t, and α is a learnable parameter. This method simply increments the current amount of water by the rain, which also seems to perform well in an area where there is already water in the domain.
- **ConvLSTM** (SHI et al., 2015): re-implementation with all CNN components with 64 channels and kernel of size 3.
- **MLP**: implementation with 3 layers with 32 neurons in each layer.
- **GAT/GCN** with 2 or 3 layers performed about the same. We kept 2 layers to reduce the number of parameters, and therefore avoid overfitting.
- **U-net** implementation with 2 down-samplings and 2 up-samplings all with 32 channels. The Swish activation function implementation from MP-PDE was used here as it increased performance.
- **MeshGraphNet**: re-implementation following description from the original paper (Pfaff et al., 2021). One layer of the proposed method seemed to perform the best, with *tanh* as the activation. For this method in particular the loss function in Eq. 11, improved the performance by a significant margin compared to L_2 loss. Note that no spatial coordinates were used in this implementation version, given the huge sizes of the domains.
- **MP-PDE** (Brandstetter et al., 2022): Adapted from the original implementation. One layer performed the best, and the prediction was conducted for one step ahead to match the configurations of other approaches used in our work. Spatial coordinates are not used like in the original implementation either. The Swish activation function was left unchanged since it performed better than *tanh* and ReLU.
- **ComGNN** showed better performance with a 3-layer MLP for Eq. 3, one layer of Eq. 8, and 2 layers of Eq. 10). *tanh* was used as the activation function and all the layers were implemented with 32 neurons.
- **FNO** (Li et al., 2021). We re-used the code proposed by the author. The best configurations that worked for us were 2 layers, each with 32 neurons, and 64 frequency modes in both dimensions. We also try fine-tuning a pre-trained version of FourCastNet (Pathak et al., 2022), FNO for weather forecasting from the same authors, but it did not perform well.

Training setup We conducted a thorough hyperparameter search on both our model and baselines and selected the configuration with the lowest RMSE score on the validation set (See Appendix A.4 for more details). The dataset has a total of 63 combinations of (watershed) regions and precipitation data, from which 9 were used as the training set, 3 as the validation set, and the remaining ones as our test set. The simulation lead time was set to 40, the largest we could train on a single NVIDIA GPU Ampere A40. For each sample in the validation set, we trained one model instance, resulting in an ensemble of 3 models per method.

A.5. Fine-grained Results

In this section, we show the breakdown of the results from Table 1 into three categories: (i) unseen regions and unseen precipitations (Table 8); (ii) unseen regions and seen precipitations (Table 10); and (iii) seen regions and unseen precipitations (Table 9). Unseen refers to the data not in the training set nor in the validation set, whereas seen means data in either the training set or validation set. We can see that our method ComGNN still performs better than the baselines.

A.6. Running Time

Machine learning models are more computationally efficient than traditional hydrodynamic models. In Table 11, we compare the running times of machine learning models (in our work) to the hydrodynamic model LISFLOOD-FP for the White Oak Bayou watershed, which has an area of 288 km^2. We find approximately a 2,000 speedup compared to LISFLOOD-FP, and small differences relative to other ML models.

Table 8: Results over both unseen regions and unseen precipitations at time step 20 and 40.

Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r\uparrow$		**CSI** $\uparrow$ $(t=40)$
	$t=20$	$t=40$	$t=20$	$t=40$	$t=20$	$t=40$	$0.001m$
Rain-Incr	.2202	.5303	.6564	.5454	.7410	.5467	.5159
ConvLSTM	.1827	.3851	.7354	.6465	.7982	.6710	.5303
MLP	.3113	.6588	.4064	.3217	.5374	.1994	.3928
GCN	.1988	.5245	.7216	.6096	.8133	.6751	.5576
GAT	.2263	.5749	.6789	.5680	.7428	.5772	.5661
U-net	.2717	.4949	.5113	.4954	.5475	.2355	.3485
MeshGraphNet	.1736	.4817	.7507	.6512	.8702	.7834	.5838
MP-PDE	.1908	.4967	.7425	.6332	.8397	.7639	.6157
FNO	.2882	.5172	.5417	.4872	.4963	.1823	.3261
ComGNN^{-}	.1718	.5051	.7332	.5108	.8648	.6279	.4707
ComGNN	**.1539**	**.3927**	**.7740**	**.6810**	**.8945**	**.8158**	**.6847**

Table 9: Results over seen regions and unseen precipitations at time step 20 and 40.

Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r\uparrow$		**CSI** $\uparrow$ $(t=40)$
	$t=20$	$t=40$	$t=20$	$t=40$	$t=20$	$t=40$	$0.001m$
Rain-Incr	.3437	.6968	.6071	.497	.6260	.5417	.5971
ConvLSTM	.2523	**.4647**	.7117	**.6422**	.7025	.6091	.6084
MLP	.4038	.7515	.4218	.3803	.4561	.3450	.4992
GCN	.3300	.7331	.6177	.5347	.6994	.6222	.6043
GAT	.3629	.7930	.5892	.5104	.6127	.5314	.6100
U-net	.4260	.6032	.4515	.4595	.4876	.3126	.4330
MeshGraphNet	.2951	.6905	.6534	.5606	.7571	.6841	.6099
MP-PDE	.3152	.7051	.6288	.5511	.7191	.6578	.6272
FNO	.3736	.6672	.5158	.4699	.4334	.2674	.4072
ComGNN^{-}	.2861	.6690	.6235	.4707	.7357	.6103	.5568
ComGNN	**.2491**	.5107	**.7204**	.6216	**.8037**	**.7363**	**.6830**

Table 10: Results over unseen regions and seen precipitations at time step 20 and 40

Method	**RMSE** $\downarrow$		**NSE** $\uparrow$		$r\uparrow$		**CSI** $\uparrow$ $(t=40)$
	$t=20$	$t=40$	$t=20$	$t=40$	$t=20$	$t=40$	$0.001m$
Rain-Incr	.2475	.6352	.5587	.5257	.6719	.5036	.7054
ConvLSTM	.2054	.5466	.6684	.6154	.7146	.6030	.7040
MLP	.3428	.7187	.3641	.4477	.4850	.3180	.6339
GCN	.2144	.5805	.6402	.5830	.7762	.6683	.7046
GAT	.2428	.6589	.6065	.5310	.6829	.5186	.7076
U-net	.3112	.6723	.4193	.5045	.5223	.3924	.5862
MeshGraphNet	.1861	.4884	.6627	.6572	.8283	.7951	.7171
MP-PDE	.1954	.5172	**.6938**	.6423	.7977	.7572	.7525
FNO	.2796	.7112	.5319	.4913	.4672	.2524	.5621
ComGNN^{-}	.1854	.5041	.6479	.6115	.8350	.7301	.6759
ComGNN	**.1674**	**.4120**	.6893	**.7151**	**.8619**	**.8406**	**.7558**

Table 11: Running time (in seconds) comparison between machine learning methods and LISFLOOD

ConvLSTM	**ComGNN**	**FNO**	**GCN**	**GAT**	**MeshGraphNet**	**MP-PDE**	**U-net**	**LISFLOOD-FP**
3.2503	2.6711	2.7830	2.6590	2.6600	2.9850	3.9243	2.3931	5511

A.7. Conservation of mass

The conservation of mass can be explained by the total change in the amount of water in the domain relative to the amount of water coming from the rain. To assess the ability of machine learning models for the conservation of mass, we take the absolute difference (error) between the change of the amount of water from the ground truth data and the change of the amount of water in the domain of machine learning models. Results are shown in Table 12. We can see that ComGNN has an absolute error of at least half of other machine learning models' errors.

Table 12: Absolute difference in total change of amount of water between ground truth data and machine learning models.

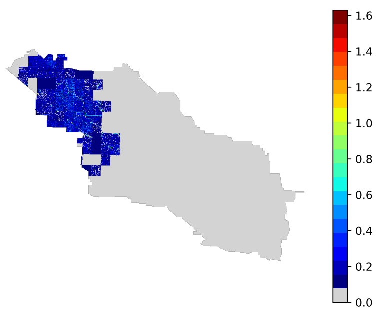
ConvLSTM	**GCN**	**GAT**	**U-net**	**MeshGraphNet**	**MP-PDE**	**FNO**	**ComGNN**
103.277	164.808	214.111	189.674	125.258	149.87	225.421	42.161

A.8. Visualization of prediction error

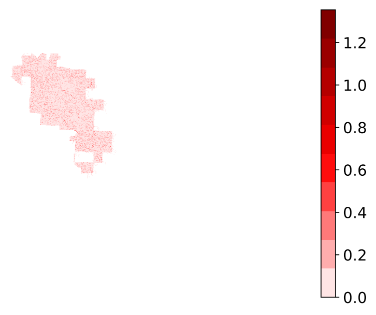
Figure 6 shows the absolute error of our model ComGNN (Figures 6b, 6e, 6h, and 6k) and the absolute error of best-performing baseline ConvLSTM, (Figures 6c, 6f, 6i, and 6l) at lead times 10, 20, 30, and 40. The first column (Figures 6a, 6d, 6g, 6j) represents the true flood map (water depths) states. We can see that ComGNN achieves lower errors than ConvLSTM. We provide further visualizations of the correlation between the true and predicted water depth values of all the methods in Appendix A.9 where we show that ComGNN's predictions are the most aligned with the true water depths.

A.9. Correlation between predictions and true values

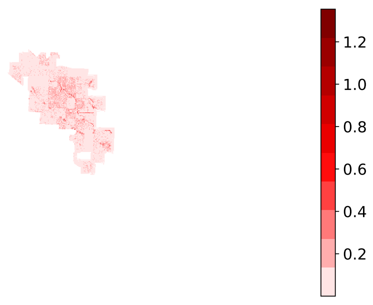
We provide visualizations of the correlation between true and predicted water depth values in Figures 7, 8, 9 and 10 for lead times 10, 20, 30, and 40, respectively. The results demonstrate that ComGNN's predictions are the most aligned with the true water depth values.



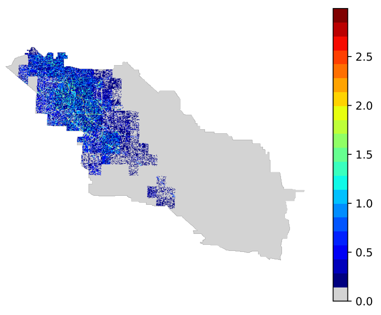
(a) True flooded area at $t_r = 10$



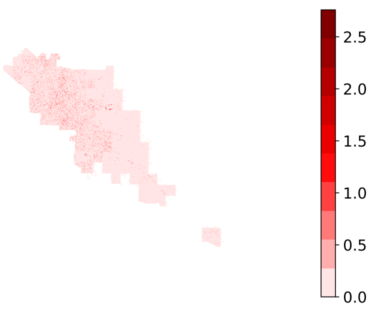
(b) ComGNN error at $t_r = 10$



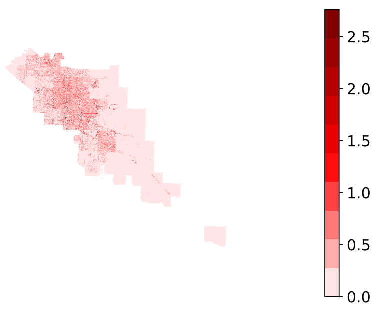
(c) ConvLSTM error at $t_r = 10$



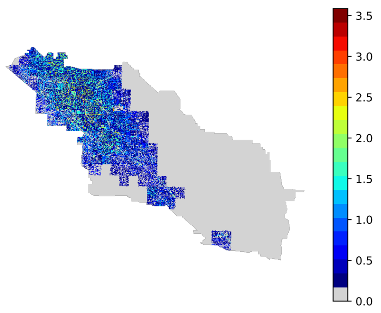
(d) True flooded area at $t_r = 20$



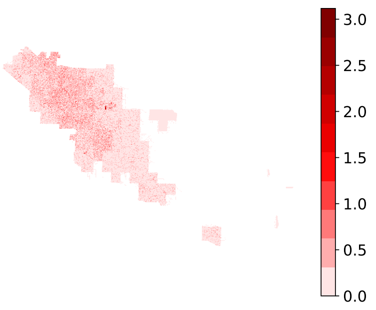
(e) ComGNN error at $t_r = 20$



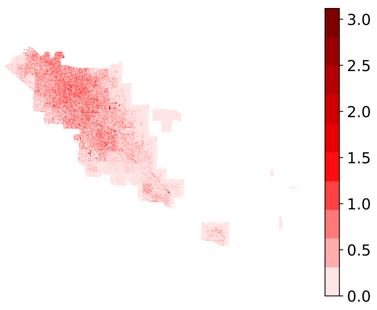
(f) ConvLSTM error at $t_r = 20$



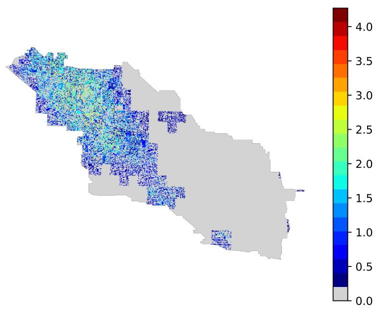
(g) True flooded area at $t_r = 30$



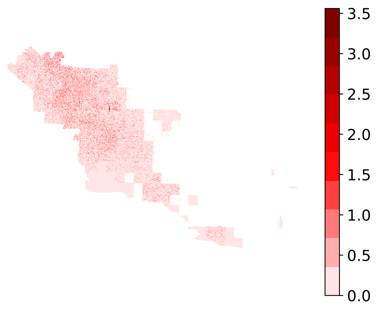
(h) ComGNN error at $t_r = 30$



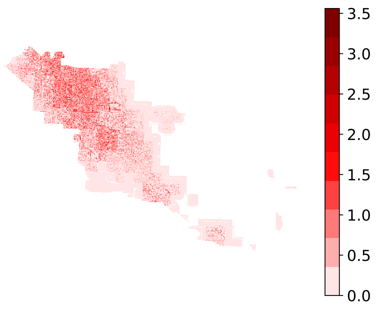
(i) ConvLSTM error at $t_r = 30$



(j) True flooded area at $t_r = 40$

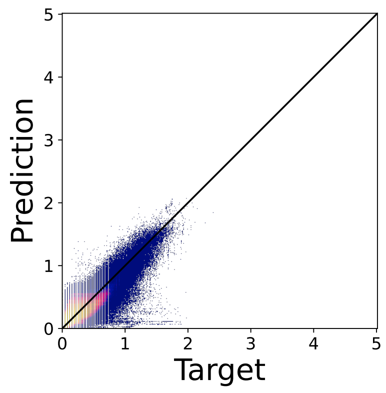


(k) ComGNN error at $t_r = 40$

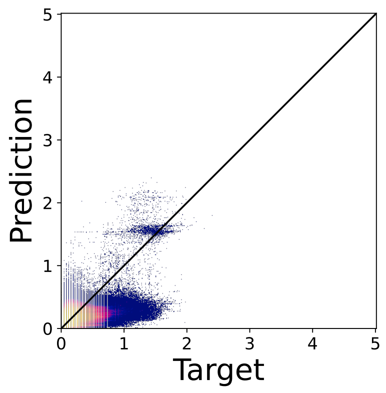


(l) ConvLSTM error at $t_r = 40$

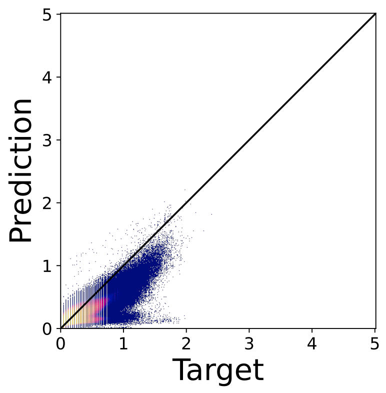
Figure 6: Absolute error of our proposed method (middle column) and the absolute error of ConvLSTM (right column) compared to the true flood area (left column) at lead times 20 (row 1) and 40 (row 2). The results show that ComGNN achieves lower error than the baseline, which is consistent with the results from Table 1.



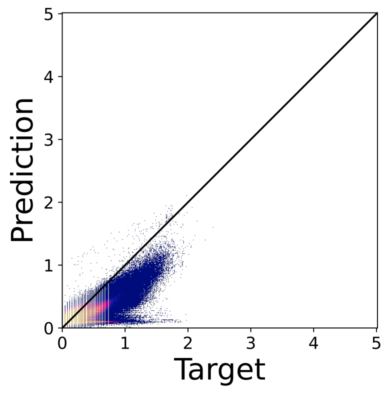
(a) ConvLSTM



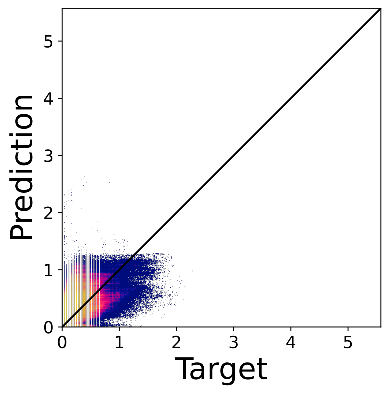
(b) FNO



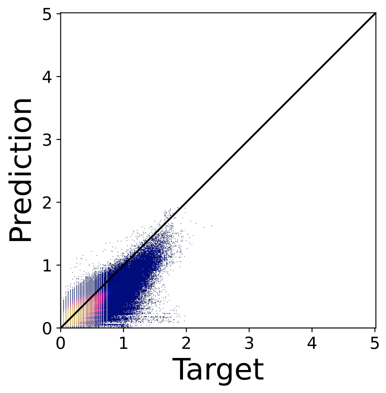
(c) GCN



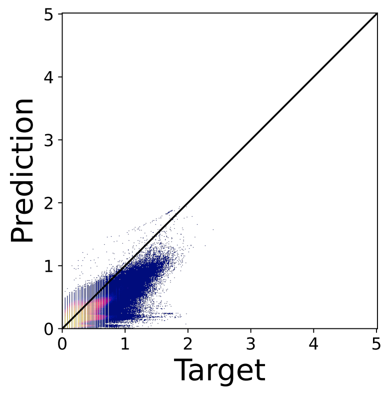
(d) GAT



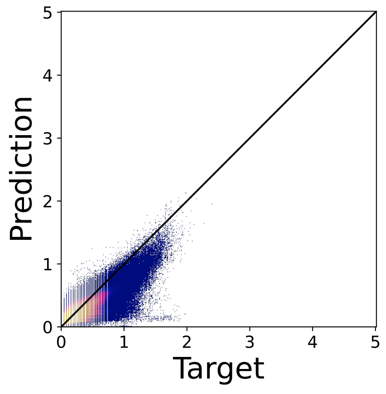
(e) U-net



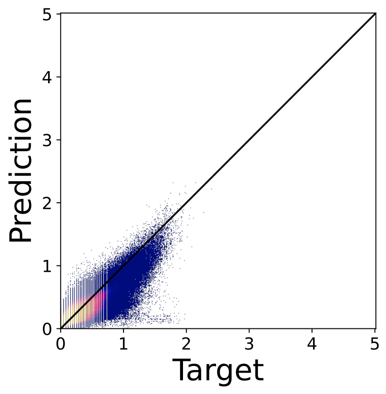
(f) MeshGraphNet



(g) MP-PDE

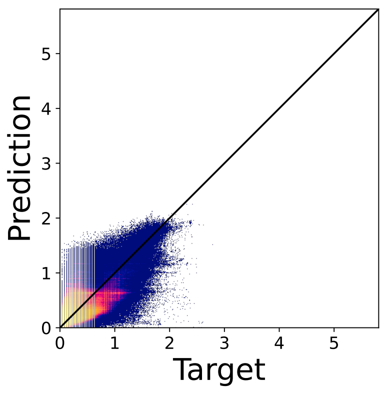


(h) ComGNN^{-}

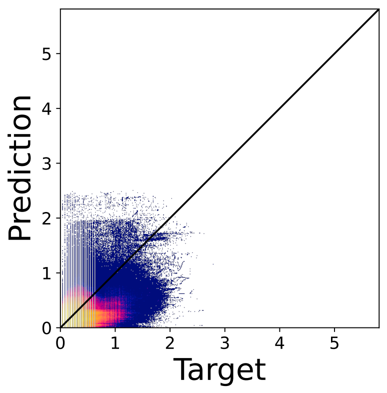


(i) ComGNN

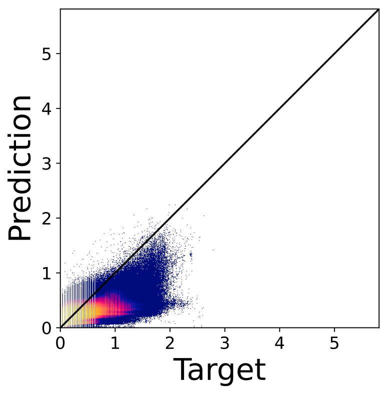
Figure 7: Scatter Plots at lead time 10 in log-log scale



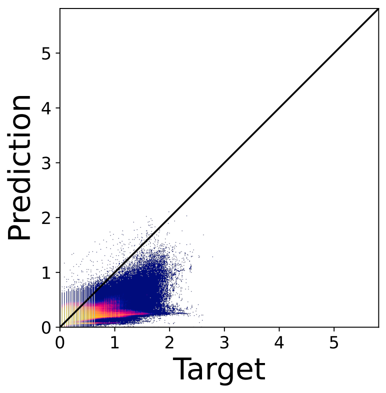
(a) ConvLSTM



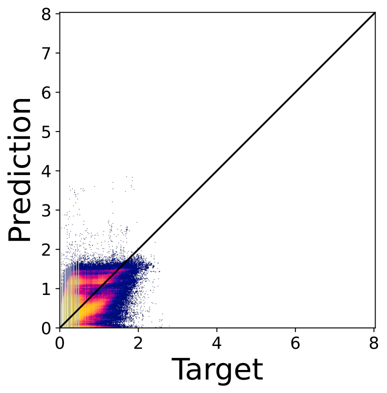
(b) FNO



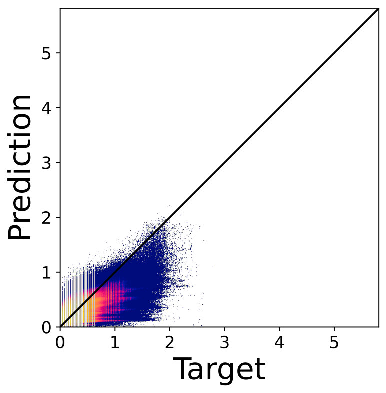
(c) GCN



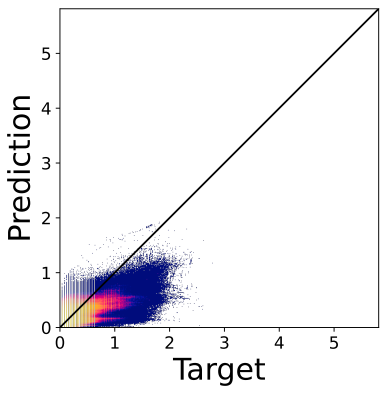
(d) GAT



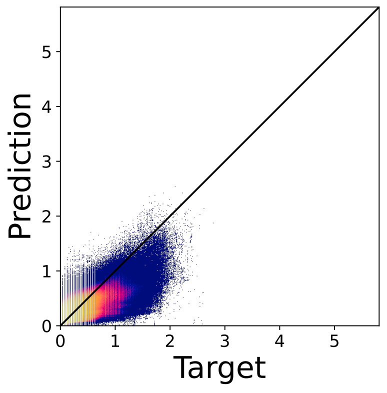
(e) U-net



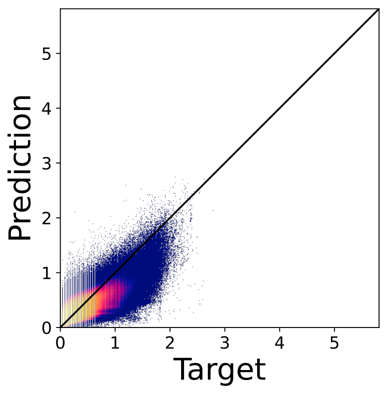
(f) MeshGrapnNet



(g) MP-PDE

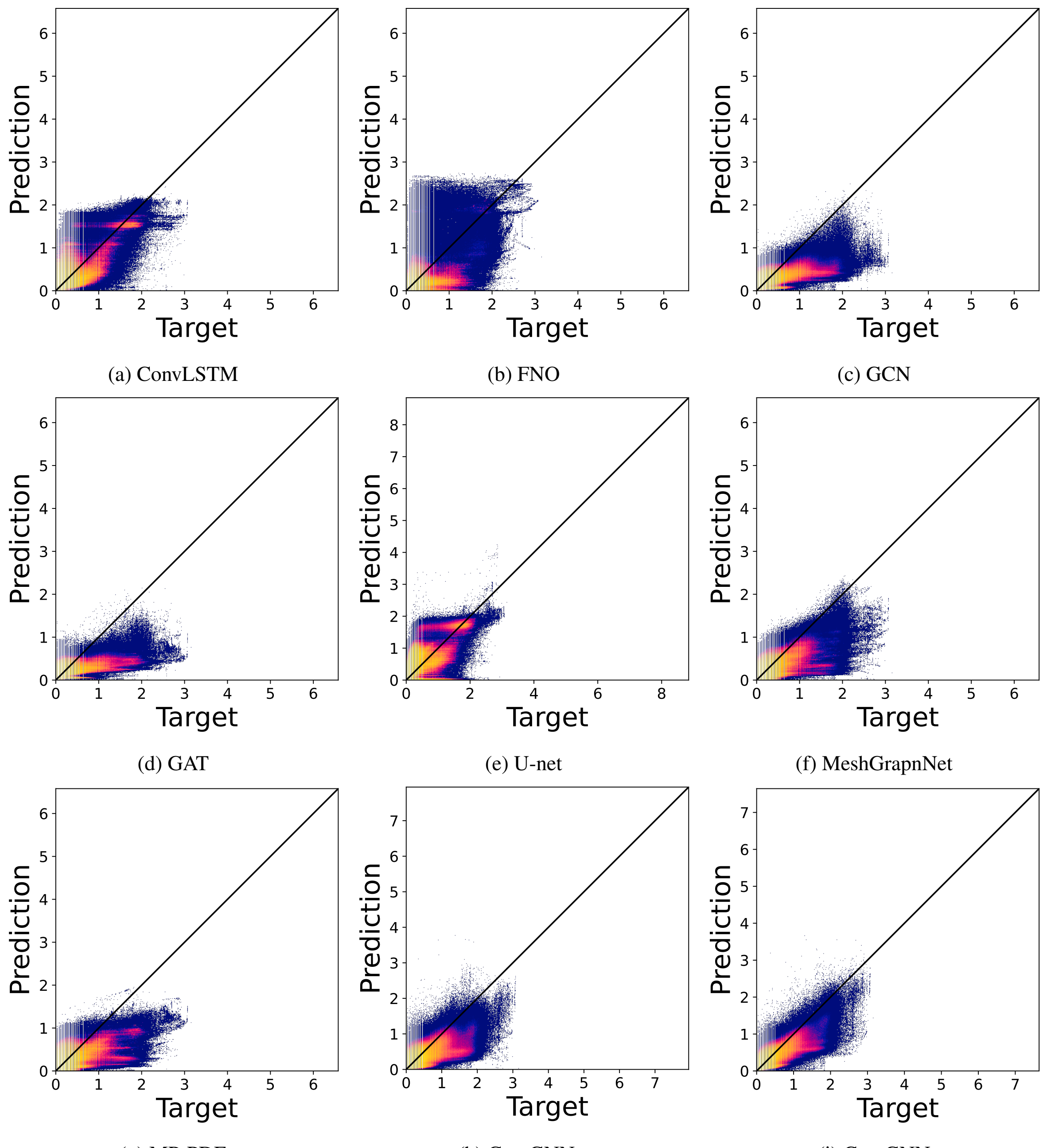


(h) ComGNN^{-}



(i) ComGNN

Figure 8: Scatter Plots at lead time 20 in log-log scale



(a) ConvLSTM

(b) FNO

(c) GCN

(d) GAT

(e) U-net

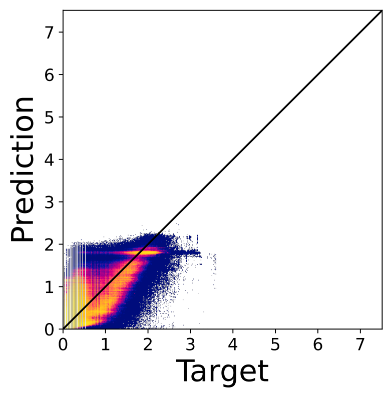
(f) MeshGrapnNet

(g) MP-PDE

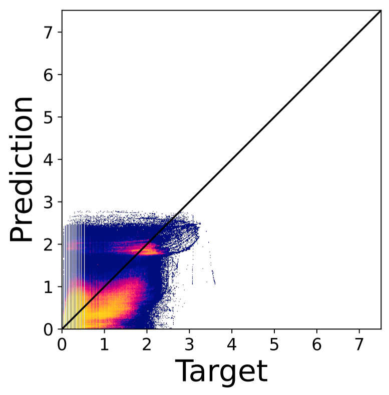
(h) ComGNN^{-}

(i) ComGNN

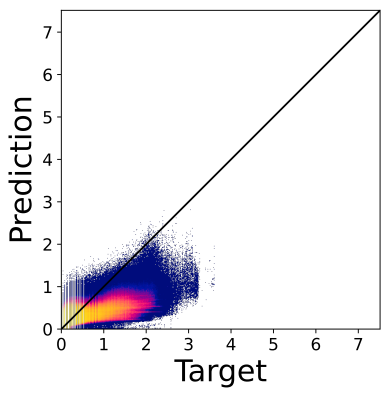
Figure 9: Scatter Plots at lead time 30 in log-log scale



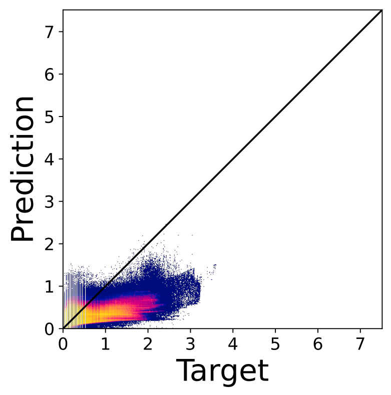
(a) ConvLSTM



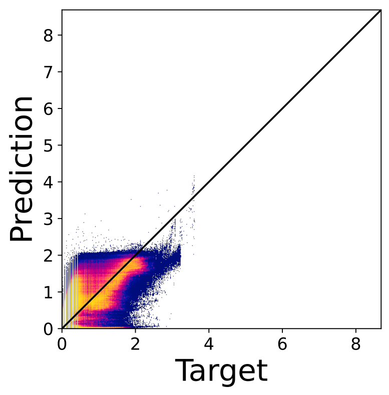
(b) FNO



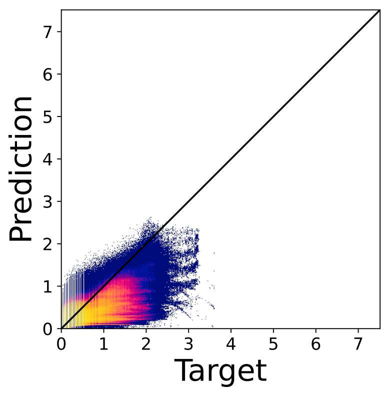
(c) GCN



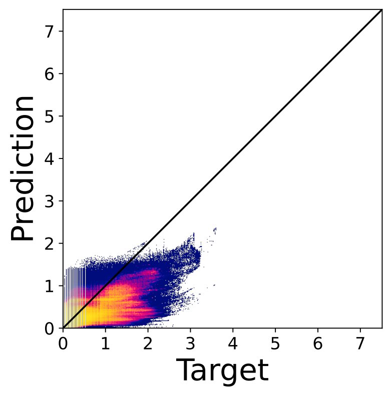
(d) GAT



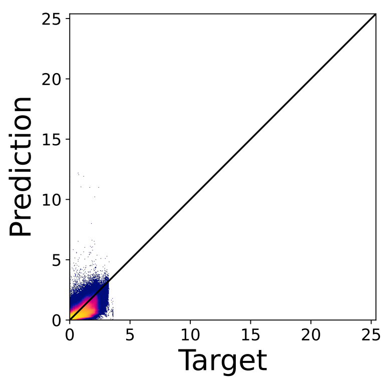
(e) U-net



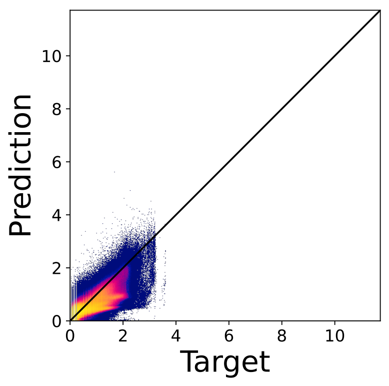
(f) MeshGrapnNet



(g) MP-PDE



(h) ComGNN^{-}



(i) ComGNN

Figure 10: Scatter Plots at lead time 40 in log-log scale