






Article

Deep Reinforcement Learning with Uncertain Data for Real-Time Stormwater System Control and Flood Mitigation

Sami M. Saliba ^{1,†} , Benjamin D. Bowes ^{1,2,†} , Stephen Adams ¹ , Peter A. Beling ^{1,2} 
and Jonathan L. Goodall ^{1,2,*} 

¹ Department of Engineering Systems and Environment, University of Virginia, P.O. Box 400747, Charlottesville, VA 22904, USA; sms8fr@virginia.edu (S.M.S.); bdb3m@virginia.edu (B.D.B.); sca2c@virginia.edu (S.A.); beling@virginia.edu (P.A.B.)

² Link Lab, University of Virginia, P.O. Box 400259, Charlottesville, VA 22904, USA

* Correspondence: goodall@virginia.edu

† These authors contributed equally to this work.

Received: 19 October 2020; Accepted: 12 November 2020; Published: 17 November 2020



Abstract: Flooding in many areas is becoming more prevalent due to factors such as urbanization and climate change, requiring modernization of stormwater infrastructure. Retrofitting standard passive systems with controllable valves/pumps is promising, but requires real-time control (RTC). One method of automating RTC is reinforcement learning (RL), a general technique for sequential optimization and control in uncertain environments. The notion is that an RL algorithm can use inputs of real-time flood data and rainfall forecasts to learn a policy for controlling the stormwater infrastructure to minimize measures of flooding. In real-world conditions, rainfall forecasts and other state information are subject to noise and uncertainty. To account for these characteristics of the problem data, we implemented Deep Deterministic Policy Gradient (DDPG), an RL algorithm that is distinguished by its capability to handle noise in the input data. DDPG implementations were trained and tested against a passive flood control policy. Three primary cases were studied: (i) perfect data, (ii) imperfect rainfall forecasts, and (iii) imperfect water level and forecast data. Rainfall episodes (100) that caused flooding in the passive system were selected from 10 years of observations in Norfolk, Virginia, USA; 85 randomly selected episodes were used for training and the remaining 15 unseen episodes served as test cases. Compared to the passive system, all RL implementations reduced flooding volume by 70.5% on average, and performed within a range of 5%. This suggests that DDPG is robust to noisy input data, which is essential knowledge to advance the real-world applicability of RL for stormwater RTC.

Keywords: real-time control; reinforcement learning; smart stormwater systems; urban flooding

1. Introduction

Flooding in many urban areas is becoming more prevalent due to changing weather patterns, rising sea levels, and increases in impervious surfaces [1–5]. The cost of stormwater flooding is estimated in the billions of dollars with the loss of property, potential for loss of life, and mobilization of emergency personnel to keep people safe [6]. Currently, the majority of stormwater systems are designed to passively manage rainfall and runoff. Although these systems are planned for urban growth and development, they often cannot accommodate the increased water volumes with new or changing construction of

buildings and roads, which decrease permeable surfaces, or the increasing storm intensities associated with climate change [7,8].

While modern stormwater systems are increasingly designed for multiple goals such as flood mitigation and water quality protection [9], they are still passive infrastructure that cannot adapt to dynamic storm events and long-term changes in runoff volumes. Retrofitting stormwater systems for active monitoring and real-time control (RTC) is a promising approach to address these issues [10]. RTC has been previously used in other urban infrastructure, such as combined sewers to prevent overflows [11]. While becoming more common with stormwater management, RTC of these systems is not yet the standard. The utilization of these dynamic systems with heuristic control has shown significant performance improvements compared to their passive counterparts [12]. Heuristic control typically offers a generalized strategy to adjust valves and pumps based on a prediction model. While shown to be successful on an individual storage pond, the repercussions of having multiple ponds in parallel are rarely considered [13]. For example, when two of these heuristic controlled systems are used, each is designed to monitor its own storage pond, and thus may cause flooding downstream by releasing water to reduce their own flooding.

As the use of stormwater RTC grows and these systems become more complex, automated control and optimization techniques are needed. Algorithmic control such as using machine learning or genetic algorithms have been proposed previously for water resource management; however, both have their faults. Machine learning has primarily been utilized to predict flooding, not reduce it [14]. Similarly, genetic algorithms have issues when the scope of the area evaluated is expanded [15]. Reinforcement learning (RL) [16], a type of machine learning, is an emerging approach to stormwater system RTC that allows the creation of policies for flow control valves, pumps, and ponds within a stormwater system using simulations [17,18]. With all forms of RL, the ultimate goal of the agent is to learn a policy (π) for the given simulated environment. A policy acts as a map for actions to a given state. In the case of this project, sensor data (state information) is connected to flow valve opening (action) at any given time. RL functions similarly in principle to the way humans or animals learn or make behavioral modifications. An action occurs and is evaluated, then a reward is applied. A greater reward incentivizes an action, and a lesser reward discourages an action. With consistent administration, the human or animal “learns” to avoid negative consequences using this technique, while positive rewards reinforce good behaviors. The RL is designed with reward functions to provide a positive or negative reward.

Deep Deterministic Policy Gradients (DDPG) is a RL algorithm that uses a deep learning framework to create function approximators with neural networks [19]. DDPG is an actor-critic RL method. The Actor and Critic are separate neural networks, but they collaborate through constant interaction. The action is performed by an actor, and the evaluation of the performance of the actor is done by a critic. The Actor takes actions in the environment based on the gradient of the Critic. The Critic receives information about the environment and “grades” the action made by the Actor. The relationship of the Actor and the Critic dictates how the overall agent learns based on a reward function that reinforces the learning [20]. DDPG is useful to learn policies in complex environments and problems in which the number of available control actions is large or a continuous space. This would make DDPG a good choice for stormwater systems in which a continuum of valve settings can be used to shift water to another location that may have upstream, downstream, or even parallel implications. The control valves used in this study can be opened to any value from 0% to 100%. Finding the most effective reward function is a key component to fully automating the stormwater systems to mitigate flooding across broad areas in a complex environment.

Bowes et al. (2020) demonstrated an improvement over the passive system, reducing flooding using DDPG when the sum of flooding was used as a reward function to train the RL agent [21]. They demonstrated that RL could reduce flooding compared to a passive stormwater system and rule-based RTC. Although they were able to reduce flooding substantially, their RL agent was trained and tested with perfect state information: the dataset used exact water levels at a current time, and the rainfall forecast was

equal to the exact future rainfall amount. In reality however, rainfall forecasts are uncertain, and sensor data are noisy.

A simulation representing the real-world must combine a multitude of sensors working in parallel, should accommodate uncertain data or imperfections in the sensors, and should account for uncertain weather conditions. A RL strategy that includes factors such as incorrect water measurements and uncertain forecasts has not been examined. Weather forecasts and rainfall prediction have poor reliability even a day in advance, and previous RTC implementations that included forecasts found that >80% of algorithmic control actions were the result of false alarms [22,23]. Rainfall uncertainty can have a significant influence on stormwater system performance [24]; for the successful application of RTC with forecasts, robustness to these variables is essential. In other applications, RL has been shown to find superior control policies despite uncertain data [25]. But the use of RL with uncertain data has not been studied for stormwater RTC. Therefore, the purpose of this study is to examine the effect of uncertain state information on the performance of DDPG in reducing flooding. We hypothesize that RL can accommodate for the uncertain data and provide an optimal or near-optimal solution, similar to perfect rainfall forecasting and fully functional sensors.

This project evaluates the effectiveness of DDPG-based stormwater control in reducing flooding based on real rainfall data from Norfolk Virginia. Expanding on previous work, we explore the robustness of a DDPG implementation by utilizing uncertain forecasting and state data. Without exact measurements, the heuristic or algorithmic control could cause catastrophic effects – holding water as the system floods, or releasing the entirety of the storage. However, this project demonstrates that DDPG can overcome uncertain data to successfully reducing flooding.

The remainder of the paper is organized as follows in Sections 2–5. Section 2 presents the methods by detailing the modeling of the stormwater environment, examining the development and implementation of a DDPG agent and RL, and describing the data. Finally, we introduce the experiment design. Section 3 presents the results comparing the RL with perfect and uncertain data to the passive system. Section 4 is a discussion of the study with further description of the utility of the design. Section 5 presents a conclusion and proposes future work.

2. Methods

This study was designed to compare the effectiveness of RL policies in reducing flooding volume under the following conditions: (i) perfect state and forecast data, (ii) perfect state and uncertain forecast data, (iii) uncertain state and forecast data. The RL policies were compared to a passive system where all stormwater valves were 100% open.

2.1. Modeling the Stormwater Environment

The US Environmental Protection Agency has developed software to simulate drainage and water transportation within urban environments called the Storm Water Management Model (SWMM) [26]. SWMM allows for the modeling of stormwater systems, taking into account runoff quality, hydrology, and elevation changes within a specific environment. It has been used when designing stormwater management systems and when development within a city changes such as building new communities or roads, as the number of impermeable surfaces affects stormwater absorption [27]. The SWMM tool allows for the modeling of a system, such as the City of Norfolk, Virginia, which commonly floods even in low-rate rainfall events when concurrent with high tides.

SWMM allows for the manipulation of specific variables such as the number of storm drains, junctions, valves, and storage areas (retention ponds) within a simulation.

Figure 1—the SWMM model utilized in this research—is based on the designs of previous machine learning implementations by Bowes et al. (2020) [21]. This model houses eight subcatchments (S_{1-8}), three major junctions (J_{1-3}), and three flow control valves—orifices—(R_{1-3}) in this system. The elevation changes are not indicated within the diagram, but are programmed into the model and the system drains downward toward the outfall (*Out*). The R valve position can be dynamically controlled in any range of openings between 0% and 100%.

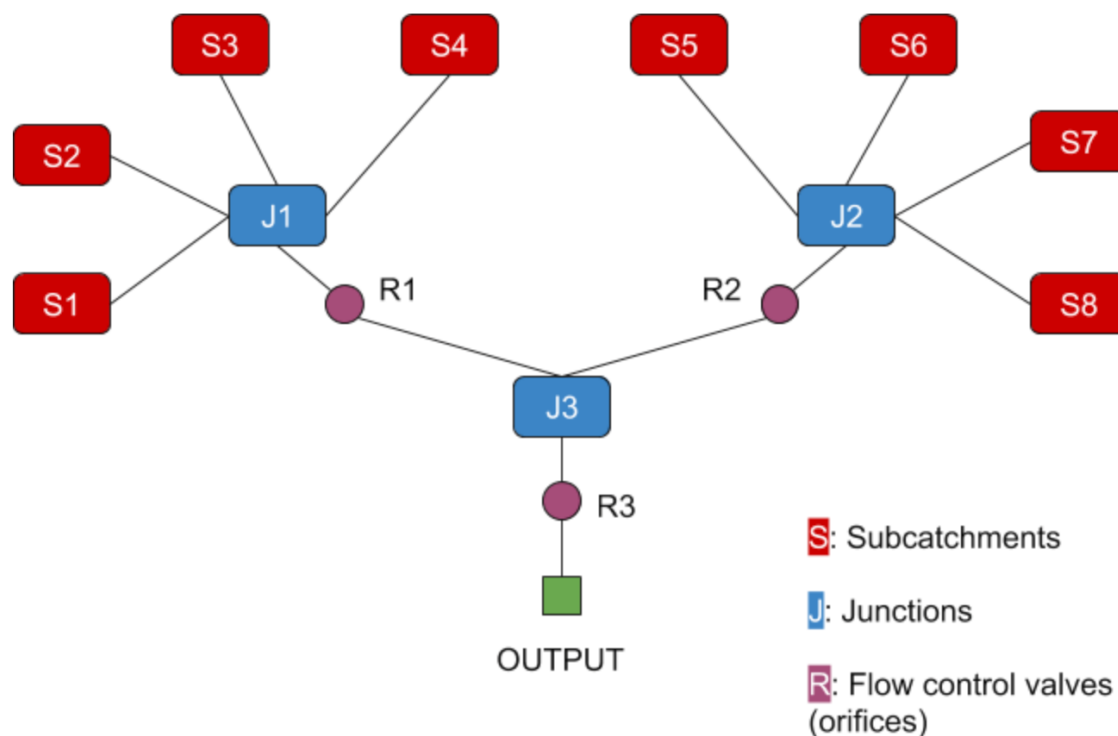


Figure 1. Graphic of the SWMM model used.

Table 1 shows SWMM setup parameters. Within the model, each subcatchment used the same rainfall data at each timestep; however, each subcatchment has unique area, impermeability, and width. Although in the table and diagram there are junctions labeled, these junctions have a secondary function as storage ponds. Throughout, they are consistently denoted as junctions; although they do have a depth component, they act as the joint output of multiple connections.

SWMM allowed for measurement of depth at each junction, accounting for water height within the system and out of the system (flooding).

Table 1. SWMM model setup parameters.

Name	Gage	Outlet	Area (km ²)	%Imperv	Width (m)	%Slope
S ₁	Gage1	J ₁	283.28	25	457.2	0.5
S ₂		J ₁	364.21	35	457.2	0.5
S ₃		J ₁	202.34	30	457.2	0.5
S ₄		J ₁	343.98	25	457.2	0.5
S ₅		J ₂	283.28	35	457.2	0.5
S ₆		J ₂	263.04	30	457.2	0.5
S ₇		J ₂	222.57	25	457.2	0.5
S ₈		J ₂	303.51	25	457.2	0.5

Name	Elev. (m)	MaxDepth (m)	InitDepth (m)	Area (km ²)
J ₁	27.4	1.2	0.2286	0.0607
J ₂	27.4	1.2	0.2286	0.0809
J ₃	25.9	1.2	0.2286	0.0809

Name	From Node	To Node	Length (m)	Roughness	Diameter (m)
C	J ₃	Out	121.92	0.01	0.2286

Name	From Node	To Node	Type	Qcoeff	Gated	CloseTime
R ₁	J ₁	J ₃	BOTTOM	0.65	NO	0.2
R ₂	J ₂	J ₃	BOTTOM	0.65	NO	0.2
R ₃	J ₃	Out	BOTTOM	0.65	NO	0.2

Name	Elevation (m)
Out	25.29

2.2. Reinforcement Learning

RL has been studied since the 1990s [16], but has only more recently become viable for complex problems with the advancement of computational technology. The sequential decision-making model for standard RL problems is a Markov Decision Process (MDP). An MDP is composed of a set of states (\mathcal{S}), a set of actions (\mathcal{A}), a reward function (\mathcal{R}), a transition function (\mathcal{P}), and a discount factor (γ). Let $s \in \mathcal{S}$ represent a specific state, and let $a \in \mathcal{A}$ represent a specific action. The reward function is generally a function of both state and action $\mathcal{R}(s, a)$. The transition function $\mathcal{P}(s'|s, a)$ defines the probability of moving to s' given s and a .

The core idea of RL is to learn from experience through an iterative process of taking actions in a given state and observing a reward and state change. The goal of the agent is to learn a policy that maximizes the sum of expected future reward

$$\begin{aligned}
 G_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \\
 &= \sum_{k=0}^{\infty} \gamma^k R_{t+k},
 \end{aligned} \tag{1}$$

where $R_t = \mathcal{R}(s_t, a_t)$ is the reward at time t .

Two key concepts in RL are the value function and the Q-function. The value function is the expected return of following a given policy starting in a state $V^\pi(s) = \mathbb{E}_\pi[G_t|s]$. The Q-function defines the value of taking an action in a state and then following a given policy $Q^\pi(s, a) = \mathbb{E}_\pi[G_t|s, a]$. An optimal policy maximizes either the value function or the Q-function. Significant advances in RL can be attributed to the integration of deep learning techniques to approximate the value function, the Q-function, or the policy.

Various types of RL algorithms have been developed to create policies for increasingly complex environments. In this case, the action is the opening or closing of the drainage valves (0–100%) within the stormwater system based on measured water height at various aspects in and around the system when rainfall data was introduced. RL allows large numbers of simulations to be run to develop a generalized policy between many different conditions within the environment. In this case, the water height or total water volume above the system (flooding) was the outcome. Thus, it was not important to pre-determine the relationship or anticipate how the drains and valves might interact. RL tests thousands of policies and iteratively performs its adjustments based on the reward function during training and testing on an unseen dataset prevents overfitting the model or biasing results. Another aspect of RL is a concept of exploration. Over time the agent may learn some policy without random exploration, but it may learn a sub-optimal policy, as the optimal may not even be discovered. To achieve this exploration, a random process generates the action of the agent for a certain amount of time during training. As training progresses, this random action becomes less common. One of the first types of deep RL algorithms was Q-learning, which allowed the creation of its own relationships among variables, rather than the programmer establishing a model [28] establishing a framework for future RL algorithms.

2.3. Deep Deterministic Policy Gradient

New RL techniques are capable of handling continuous states and actions, a requirement for this project. Deep Deterministic Policy Gradient (DDPG) is an Actor-Critic RL algorithm that combines some of the optimal characteristics of deep Q learning in a continuous action space. Although this algorithm was developed in 2015 and has been tested primarily on games—Atari games are the current industry standard for evaluating reinforcement learning [19]—DDPG has potential applicability for large, complex systems such as urban infrastructure because of its ability to integrate the continuous and streaming datasets with interactive variables. Interactions between SWMM and DDPG are processed through an interpreter [29], converting DDPG output into a context readable by SWMM and vice versa with SWMM output. This interpreter is loosely based around the OpenAI Gym environment [30], a standardized framework for RL and environment interaction and RL is implemented with the keras-rl python package [31]. The python PySWMM wrapper for SWMM allows for the SWMM model to be run incrementally, an essential requirement for RL [32].

DDPG can be divided into two major parts: The actor network, $\mu(s|\theta^\mu)$ is a function which updates the current policy π mapping states to actions. The critic, $Q(s, a|\theta^Q)$ learns using the Bellman equation utilized by past RL such as Q learning [28] (see Figure 2).

The actor and critic are initialized with weights θ^μ and θ^Q . With this, the target networks of Q' and μ' are also initialized with the weights of Q and μ creating $\theta^{\mu'}$ and $\theta^{Q'}$. Due to the constant updating of the actor and critic neural networks, and the actor being based on the critic's network, target networks are utilized to slowly update the networks. A replay buffer (\mathcal{B}) is also initialized. \mathcal{B} stores state, reward, and action information as the episode progresses and is sampled for training. \mathcal{B} and target networks improve the stability of DDPG as one imperfect event will not cause a drastic change to the model.

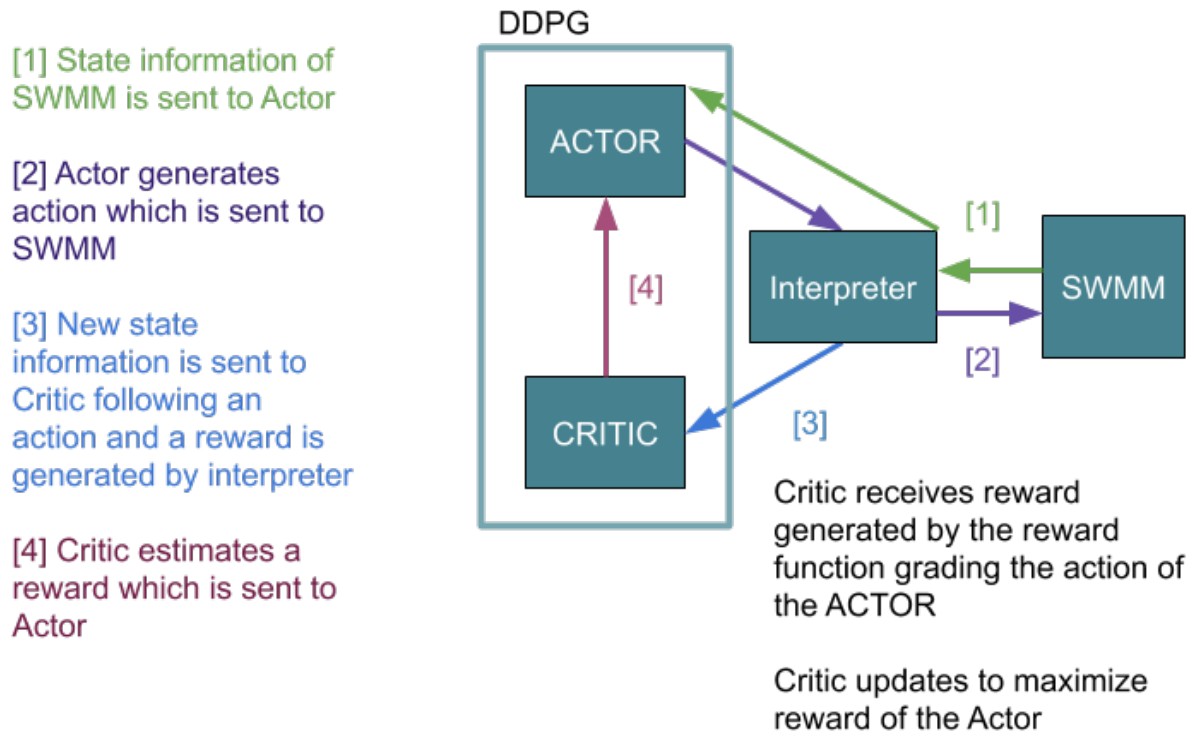


Figure 2. Communication between SWMM and DDPG.

The simulations begin with an initial observation and a random process $\epsilon \sim \mathcal{N}(0, \sigma)$ for action exploration. As the simulations progress, an action is selected at each time step using the formula,

$$a_t = \mu(s|\theta^\mu) + \epsilon. \quad (2)$$

Then, for each step in the simulation: The action is performed, resulting in a reward, and the next state, s_{t+1} is observed; all stored in \mathcal{B} .

As training begins, a random batch of s, a, r , and s_{i+1} of size (N) is sampled from \mathcal{B} . Intermediary variable y_i , which is the expected future reward as predicted by the target networks, is created,

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}))|\theta^{Q'}. \quad (3)$$

The critic network is updated to minimize the loss; the sum of the difference between y_i and the critic with respect to the current state and actions where i represents the i th sample,

$$L(\theta^Q) = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2. \quad (4)$$

The actor policy is updated using the sampled policy gradient of the critic with respect to the actions multiplied by the policy gradient of the actor with respect to the state. This is then averaged by the total size of the batch,

$$\nabla_{\theta^\mu} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \cdot \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}. \quad (5)$$

The weights of the target networks are then updated, discounted by learning rate multiplier τ ,

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \quad (6)$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}. \quad (7)$$

This process iterates for the duration of the simulations, starting with parameter, $t = 0$

2.4. MDP

The DDPG functioned on the following MDP state-action space:

1. The state is composed of several features defined below:
 - the depth of the water at junction $i = \{1, 2, 3\}$ is defined as d_i ,
 - the amount of flooding at junction $i = \{1, 2, 3\}$ is defined as f_i ,
 - the current orifice setting at junction $i = \{1, 2, 3\}$ is defined as o_i ,
 - the predicted rainfall within the current hour is defined as r_1 , and the predicted rainfall within the next hour is defined as r_2 .

$$s_t = [d_1, d_2, d_3, f_1, f_2, f_3, o_1, o_2, o_3, r_1, r_2]. \quad (8)$$

2. The action: at the next time step set valve position o_i to value $x \in [0, 1]$
3. The reward: $\mathcal{R}(F, D, E)$, where $F = f_1 + f_2 + f_3$, $D = d_1 + d_2 + d_3$, and E is the target depth of 0.6096

$$\mathcal{R}(F, D, E) = \begin{cases} -F^2 \cdot [\sigma(f_1), \sigma(f_2), \sigma(f_3)] & F > 0 \\ |D - E| & F = 0 \end{cases}$$

where σ represents the sigmoid function:

$$\sigma = \frac{1}{1 + e^{-x}} \quad (9)$$

As a starting parameter, $o_i = 0.5$.

\mathcal{R} is a piecewise function, with flooding; the reward is the dot product of the average of the sigmoid of the flooding at different junctions multiplied by the negative sum of flooding. The sigmoid function utilized normalizes the total flooding amount at each node in a range of zero to one. This would cause the reward to be approaching the actual flooding value as flooding increases. An even more general multiplier for each node was created as the episode progresses due to averaging. In other words, even if flooding was very large for a short period of the episode—potentially indicative of a flash flooding event—averaging it would not as negatively penalize the RL for unchangeable externalities such as an amount of rainfall that would greatly overwhelm any system, passive or controlled. If there is no flooding, then the reward is dependent on the difference of water level from the expected height.

A secondary addition to the function is a minor reward related to the water depth and its difference from an expected depth (E) of 0.6096 meters, 50% of the junction max depth. In the context of real-world applications, these junctions/storage ponds are not meant to run dry. As a result, this expected height would act as the intended safe operating depth for the pond. Application of this solution to a real-world situation would require an examination of the results to promote greater distribution of water over the entire system where the total water volume could be managed. For the reward function, the sum of the total flooding F , and depths D are used.

After pilot testing, a reward function was selected for further training and testing. Pilot testing included five total rewards each evaluated against the passive and each other following 10,000 training steps with the same setup as this study. It was determined that a beneficial reward function should include a dot product of an array method to prevent a policy that placed all of the flooding on one node. Although the total volume of water might be less, it would be a sub-optimal policy since one node would have a catastrophic amount of flooding.

2.5. Data

For experimentation, a rainfall data set was obtained from two gauges operated by the Hampton Roads Sanitation District in Norfolk, Virginia (Figure 3). This data set recorded rainfall in 15-min increments from 2010 to 2019. As this is a generic simulation loosely representing a stormwater system in Norfolk, Virginia, the mean of the two data sets was used to provide a rainfall time series more representative of the larger area. From this data, 100 episodes in 24-h periods were selected that were found to result in flooding using a passive system. This was done in order to reduce variability within the successes of the RL implementation—if the RL agent does not have to make any actions and no flooding occurs, it takes longer for an optimal policy to be found.

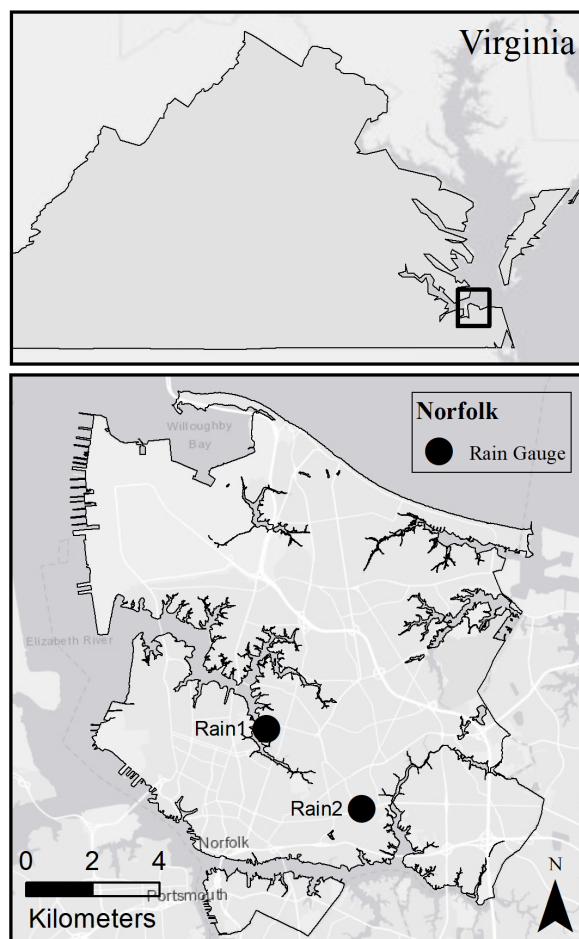


Figure 3. Map of the Norfolk area with rain gauge locations.

2.6. Experimental Design

Training simulations were run 10,000 times, making measures and adjustments on a 15-min basis, with each simulation corresponding to a 24 h period within the data set. Training was repeated with each of the testing conditions. This means that in total, 240,000 h were simulated for each condition, equating to a total of 16,000 15-min periods.

For each condition, the episodes were processed into a simulated perfect forecast, giving the sum of the depth of rainfall received during each period in one-hour increments. In other words, one value was created for each hour to represent the total rainfall for that particular hour. This was done for the current hour of the simulation, (f_1), and the next, (f_2).

Forecasts were found to have an approximate 40% accuracy within a tertile of variation six days in advance; however, the accuracy greatly increases as the timescale decreases (1 h in advance increases accuracy) [33]. The perfect forecast generated has no imperfections, as such to evaluate more real-world performance, noise is added. Thus to simulate uncertain forecasts, (f'_{1-2}), a uniform distribution (U) was sampled in order to generate a value between the upper (A) and lower bounds (B) of the accuracy,

$$X \sim U(A, B). \quad (10)$$

This uncertainty was simulated by the product of the perfect forecast and sampled value (X) with upper and lower bounds of 0.95 and 1.05 to create a possible spread of values in between $\pm 5\%$.

$$f'_1 = f_1 \times X \sim U(0.95, 1.05) \quad (11)$$

The bounds of the sample became greater the further the forecast was from the current timestep; an example being $\pm 10\%$ two hours in the future.

$$f'_2 = f_2 \times X \sim U(0.90, 1.10) \quad (12)$$

During the same time period, the SWMM tool simulated water heights and flooding at each node.

For the perfect state data, the exact sensor and exact forecast data were used. For both the uncertain state and uncertain forecast data, the uniform sampling discussed above was utilized; however, additionally, a uniform distribution was again sampled $\pm 10\%$ for the state data to create the uncertain state condition.

A random sample of 15% of rain events was selected for test episodes (rainfall test episodes 1–15) to compare the performance of each trained RL to that of the passive condition. These test episodes were not used for training to prevent and test for overfitting—the policy is only optimal for the training data. The passive condition mimicked the traditional stormwater systems which are commonly used, and served as a control condition for the RL implementations.

3. Results

3.1. Overall Comparison of RL Agents and Passive System

The performance of each DDPG implementation with perfect and noisy forecast and state information was evaluated against the passive system for the 15 randomly chosen test datasets (Table 2). The table lists the total flooding volume for each episode. Each implementation showed major reductions in overall flooding volume compared to the passive system (Figure 4). When compared to the passive system, total flood volumes had average reductions of 69.23%, 67.90%, and 73.02% for the RL implementations using perfect data, noisy forecast data, and noisy state data, respectively. The episodes 1, 5, and 10 were randomly selected to further explore the performance of the implementations on a step basis.

Table 2. Total flood volumes and percent change compared to the passive system for the 15 test events from each control method. Testing episodes which were further evaluated are in bold.

Testing Episode	Passive (m ³)	Perfect RL (m ³)	Noisy Forecast RL (m ³)	Noisy State RL
1	9,227,798	6,279,419	4,203,413	4,545,268
2	2,086,213	0	176,732	0
3	3,334,000	0	195,665	0
4	502,718	0	90,518	0
5	5,982,981	207,619	0	0
6	1,785,434	0	135,982	0
7	2,258,397	0	159,199	0
8	7,035,674	337,263	0	0
9	1,232,353	0	250,331	0
10	23,302,974	19,503,673	18,023,794	17,336,890
11	20,115,112	18,056,242	16,624,101	16,936,806
12	20,265,776	13,885,644	12,774,289	12,496,106
13	614,207	0	89,129	0
14	11,353,640	4,223,142	3,803,721	4,415,311
15	18,299,270	12,509,538	12,183,680	11,279,106

Passive Comparison	Perfect RL	Noisy Forecast RL	Noisy State RL
Mean % reduction	69.23%	67.90%	73.02%
Median % reduction	96.53%	81.99%	100.00%
Standard deviation % reduction	36.35%	29.34%	32.77%

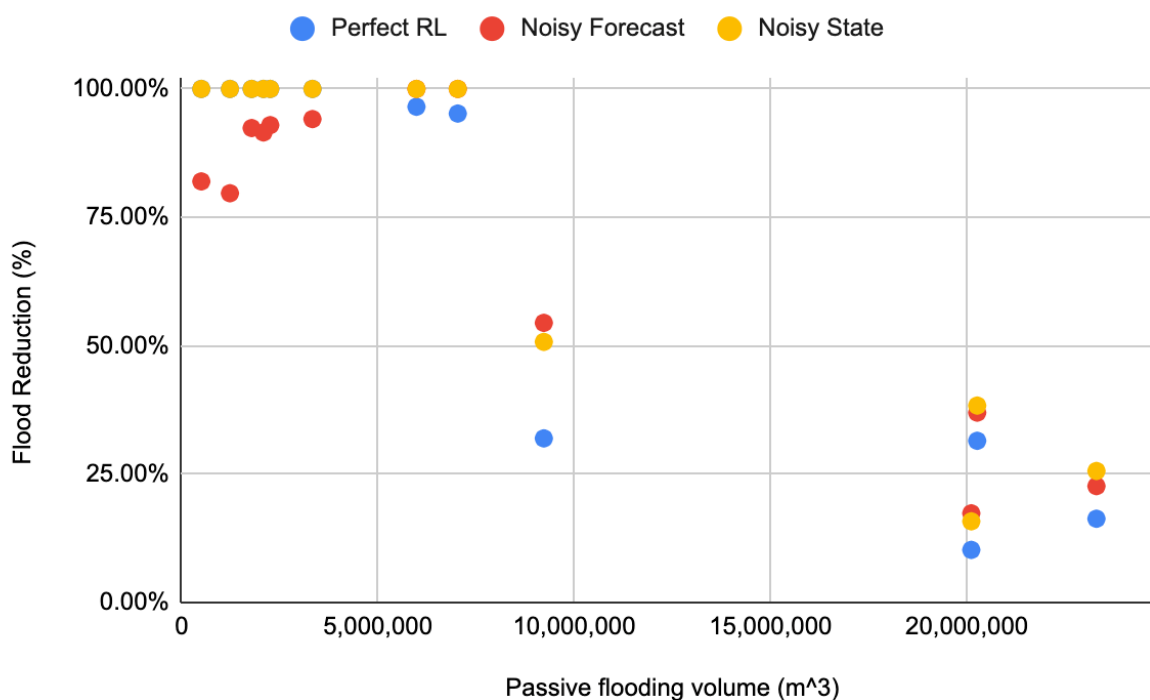


Figure 4. Percent reduction of flooding volume of the RL implementations compared to the passive condition. The similar performance of the implementations result in the range of the mean % reductions to be within 4%.

3.2. Analysis of RL Agent Policies

Of the 15 test rainfall events used to evaluate the performance of the RL implementation, three events—1, 5, and 10—were randomly selected as a sample for further analysis. The rain events for testing episodes one, five, and ten are displayed in Figure 5.

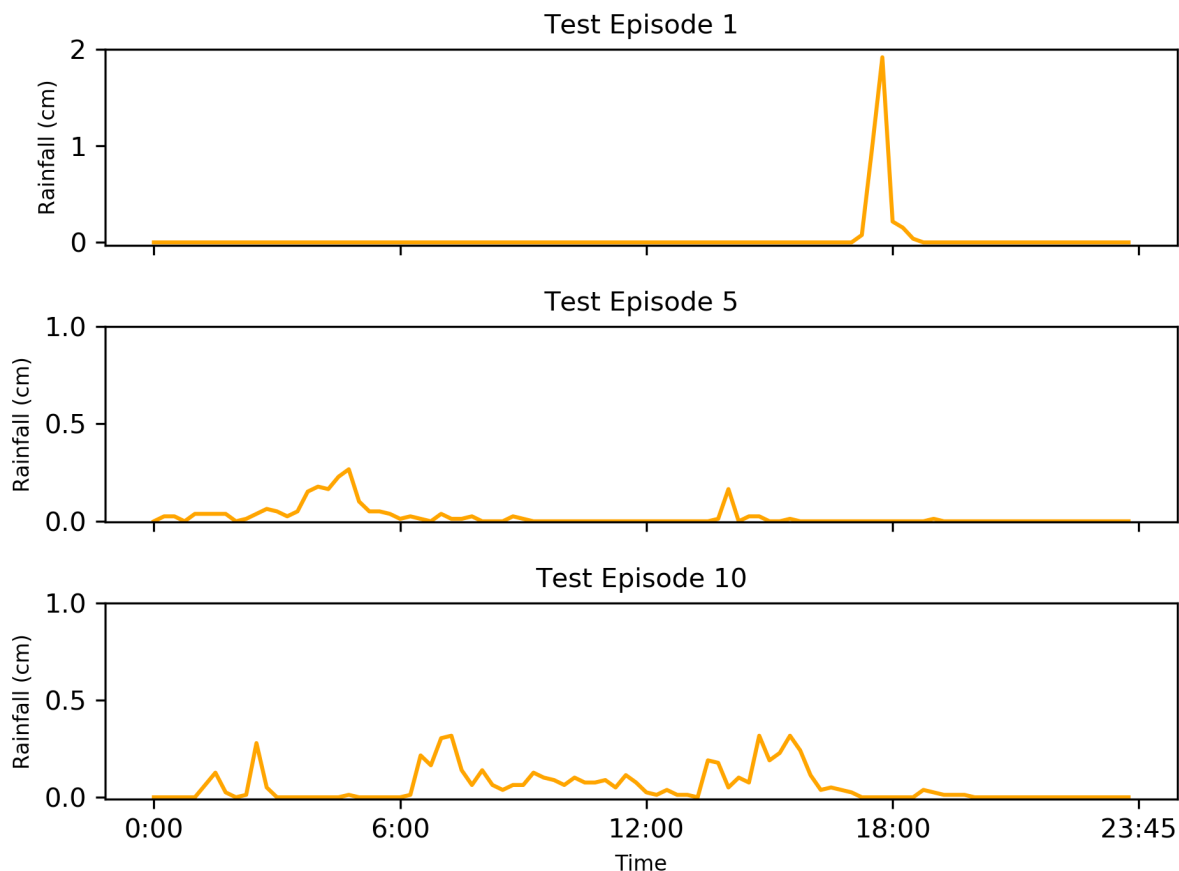


Figure 5. Rainfall events further analyzed: Test Episode 1—short duration high intensity storm peaking at 1.9177 cm of rainfall in a 15 min period. Test Episode 5—long duration low intensity storm peaking at 0.2667 cm of rainfall in a 15 min period. Test Episode 10—long duration low intensity storm peaking at 0.3175 cm of rainfall in a 15 min period.

For test episode 1 (Figure 6), where there was approximately 2 cm of rainfall within a 15 min period, all stormwater implementations were quickly overwhelmed. The passive system resulting in $9 \times 10^6 \text{ m}^3$ of flooding, performed the worst. As all of the valves are open, J_1 and J_2 do not see substantial volume; however, J_3 is quickly overwhelmed resulting in the large flooding amounts. The RL implementation using perfect data was able to modulate the valves, resulting in a 32% reduction. Opposed to the passive system, the perfect RL caused flooding at J_1 and J_2 while keeping J_3 below flood stage. In a real-world context, while still causing flooding, it would result in less damage distributing the flood volume across two different areas as opposed to $9 \times 10^6 \text{ m}^3$ at one. Both of the noisy data variants performed similarly resulting in a 50+% reduction. Similar to the perfect data implementation, both kept J_3 below flood stage, but they achieved this—noisy state especially—by rapidly cycling the orifices from 0–100%. In a simulated environment, this may be optimal, but in a real-world context, this may not be a desirable behavior. Further research, potentially with another reward function, would be needed to avoid this behavior.

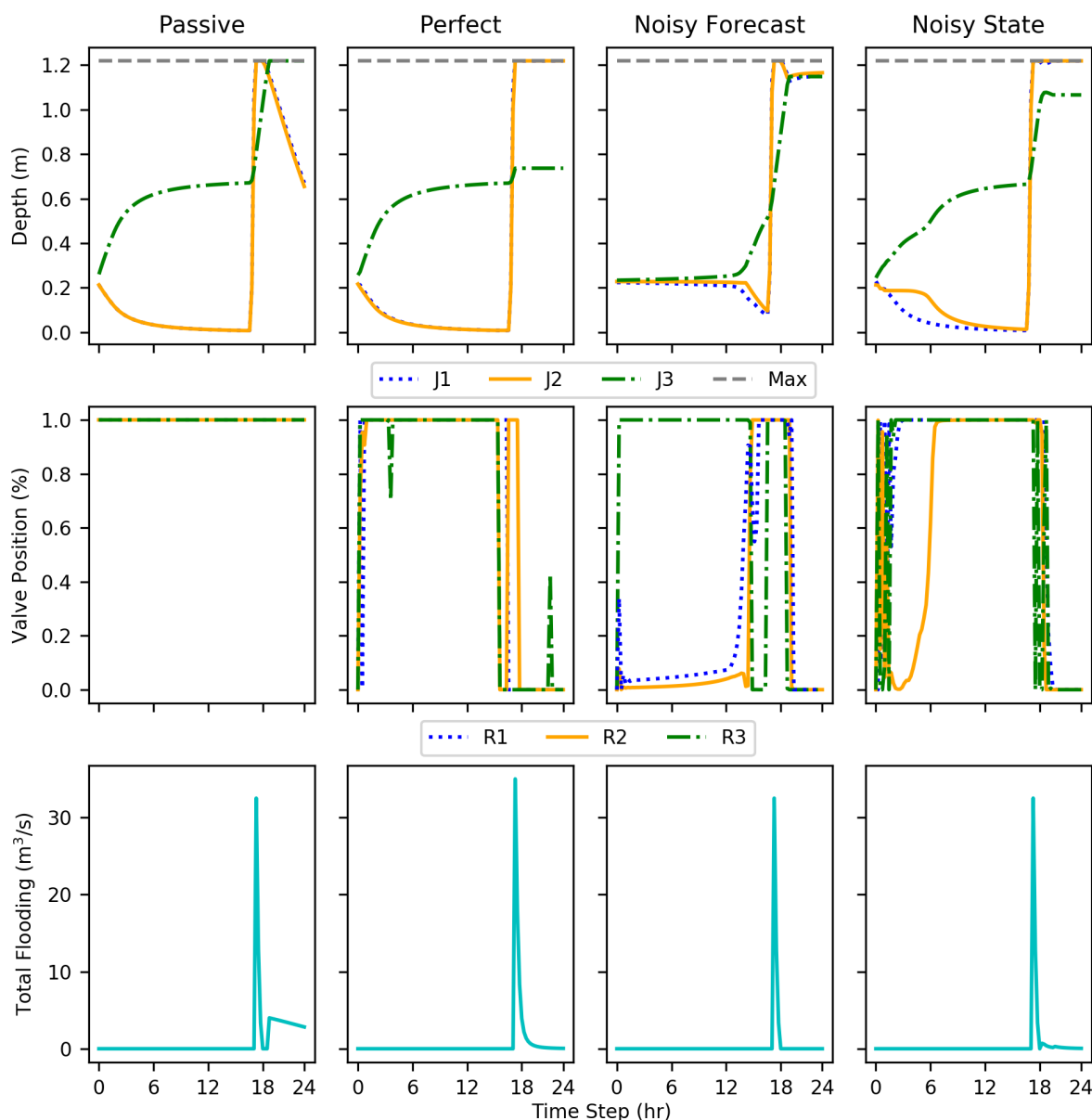


Figure 6. Comparison of the passive system to the RL implementations (episode 1). Each implementation performed better than the passive condition. In this instance the Perfect RL performed worse than the other two implementations only resulting in a 32% reduction opposed to the ~50% reduction of the other two.

For test episode 5 (Figure 7), there was peak rainfall of approximately 0.4 cm of rainfall over a 15 min period, however there was a continuous, yet minor rainfall for 24 h period. While a seemingly small quantity of water, multiplied across the area of eight subcatchments, the perfect implementation was overwhelmed. Similar to the first episode, J_{1-2} were well below flood stage; however, J_3 flooded. In the case of this episode, flooding occurred for nearly 12 of the 24 h resulting in $6 \times 10^6 \text{ m}^3$, the passive performing the worst. The perfect data implementation was able to modulate the valves, resulting in a 96% reduction. Similar to the first test episode, the perfect data implementation caused flooding at J_1 and J_2 while keeping J_3 below flood stage; again distributing the flooding volume to two areas. Both of the noisy

data variants performed similarly resulting in a 100% reduction. Each case was able to keep each junction below flood stage.

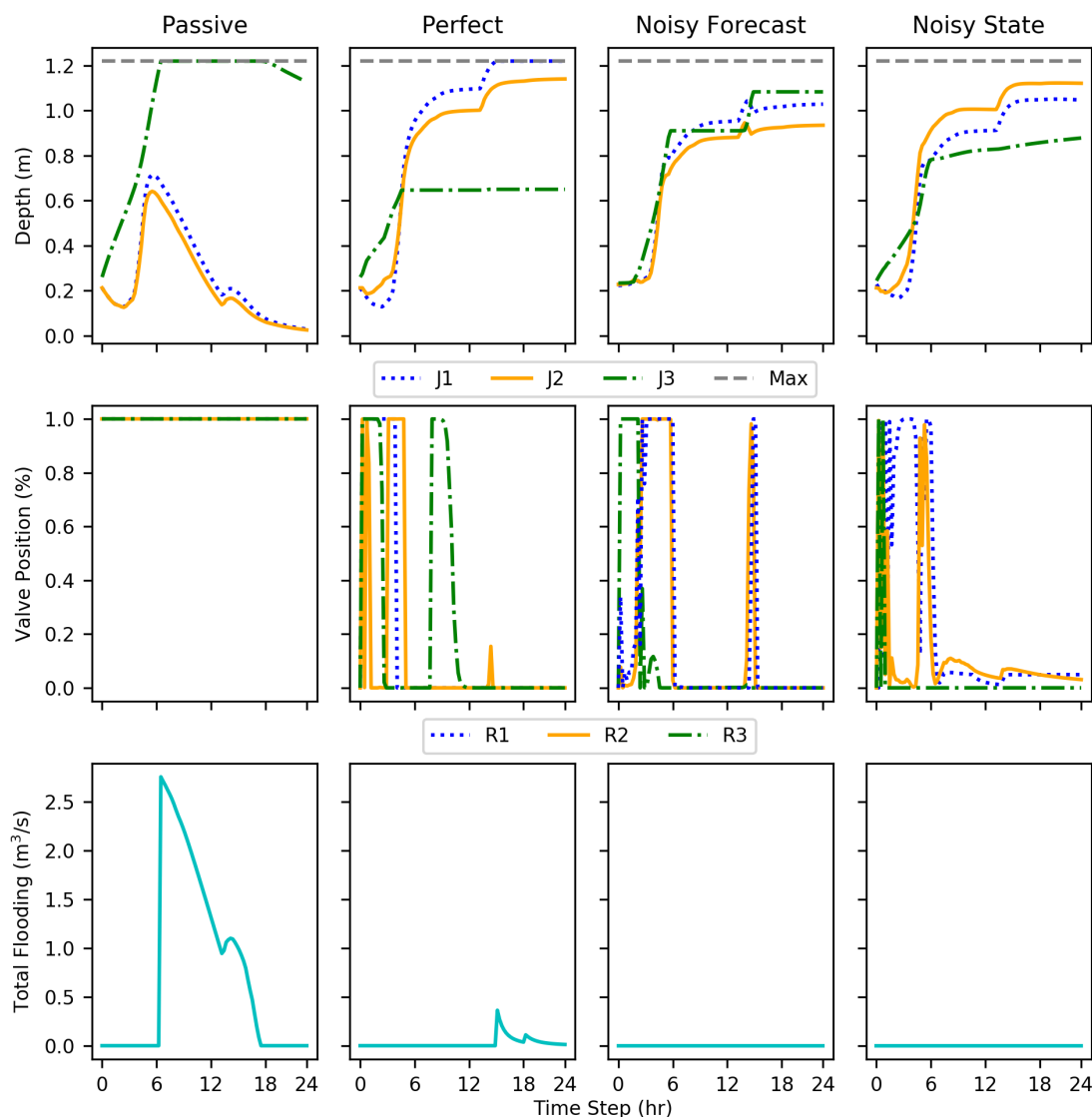


Figure 7. Comparison of the passive system to the RL implementations (episode 5). All implementations are able to reduce flooding, yet reducing a lesser amount than the first test episode.

For test episode 10 (Figure 8), there was a moderate intensity storm peaking at 0.3 cm per 15 min. The storm fluctuated around that volume for the duration of the 24 h. The passive system resulting in over 23×10^6 m³ of flooding, performed the worst. All junctions had flooding, J_3 the most. Opposed to the other test episodes, all of the RL implementations resulted in flooding. The junctions are all overwhelmed in a similar way, yet minor differences result in differences in flooding volume. The perfect data implementation preforms the worst with a 16% reduction, and the imperfect state resulting in a 25% reduction, a range of 9%.

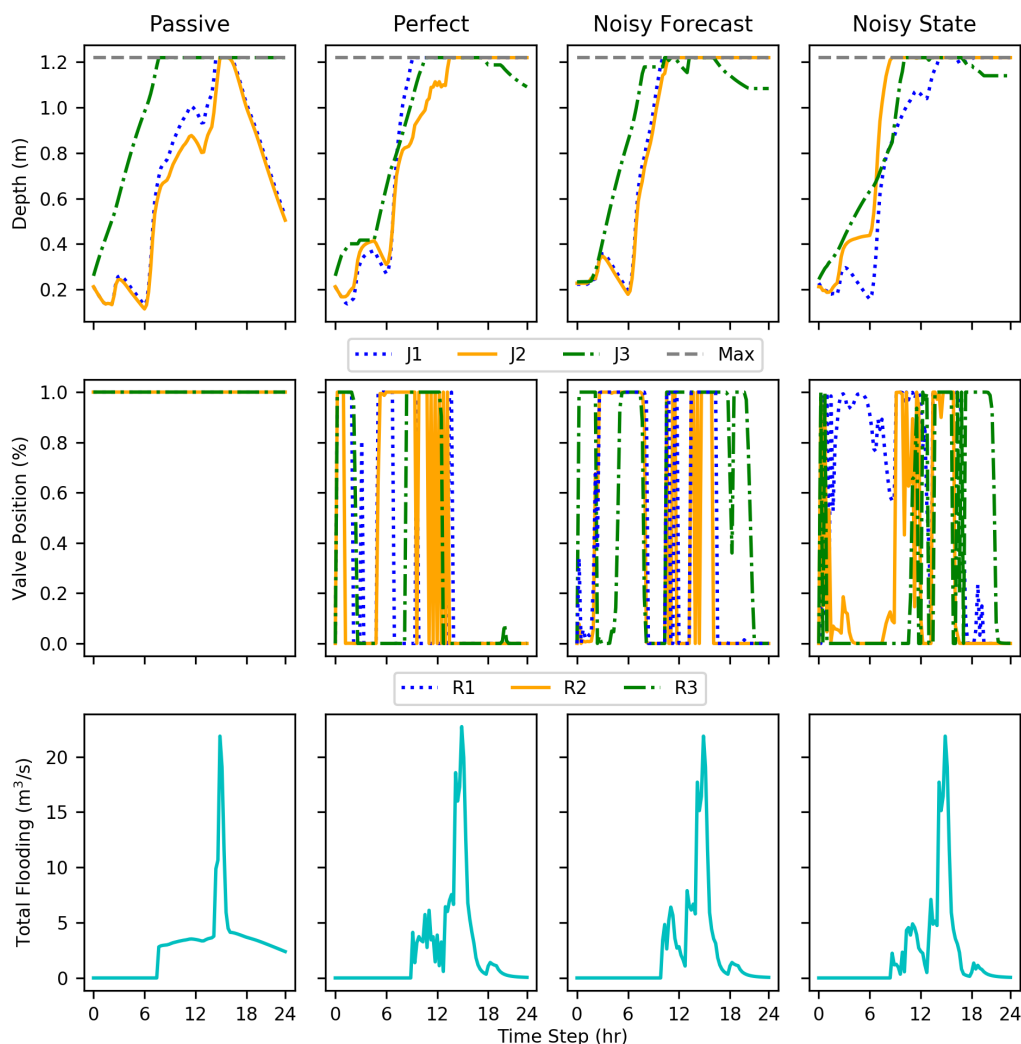


Figure 8. Comparison of the passive system to the RL implementation (episode 10). With this test episode, each implementation shows rapid cycling between 0 and 100% open. All show a reduction compared to the passive condition.

4. Discussion

The DDPG implementations, even with added noise, showed reductions in flooding compared to the passive conditions, evaluated on 15 test conditions. The tests show that DDPG learned a policy that reduced total flood volume by an average of 69–73% compared to the passive system and several test events had 100% reductions. The implementations were evaluated using disadvantages more realistic in real-world conditions such as uncertain forecast and state data. Nevertheless, each DDPG implementation performed similarly to the others. This shows the ability to account for and even learn how to use data formerly unused due to unpredictability and uncertainty, such as forecasting.

Within the results, each implementation at some point showed rapid cycling between valve positions; this is particularly evident with test episode 10 (Figure 8). This action may be indicative of incomplete training, a fully optimal policy not being completely realized.

While only resulting in an average decrease in flooding volume by an additional one percent, the noisy state implementation performed better than the perfect condition. Without more extensive testing,

any explanation may be speculation, but, the better performance can most likely be attributed to further exploration of the state space due to increased irregularity with the state information. Another explanation for the irregularity is simply variation within training. While ideally an optimal policy can be learned every time, there is an aspect of random exploration to RL.

This project acted as a first step showing that DDPG can perform optimally even with uncertain data. However, for an implementation such as this to be utilized in the real world, there are still a multitude of factors that need further exploration, such as the consequences of control system or sensor failure. While uncertain data are more representative of reality and can often lead to unpredictability within a system, the loss or removal of data can have even graver consequences. If a human were to look at operating levels of something such as a storage pond, the human could realize highly improbable values such as a depth of zero, but a machine without vast training on a wide array of issues may never realize there is an error. RL or DDPG solutions are currently limited by what can be simulated. Without the knowledge of what can fail and how, one can only speculate as to the ramifications of a system continuously flooding a storage pond as it thinks it is below standard operating height due to sensor failure.

Reinforcement learning is an evolving field, and computer systems are able to handle larger datasets. A multitude of factors go into the performance of a reinforcement learning implementation, DDPG being no exception [19]. This study utilized a reward function selected following pilot testing of 5 others, each using different methods of administering a negative reward. Other formulae may have produced less total flooding volume, but potentially may be catastrophic based on the concentration of where the flooding occurred. However, without further area specific evaluation; this remains speculation. The reward function chosen for further training and testing would mitigate flooding, but it may have unforeseen consequences based on the current infrastructure model. Additional training cycles would likely improve the results.

The reported rainfall amounts and subsequent flooding, although simulated, were representative of historical events in Norfolk, Virginia. Norfolk has been especially affected by changing weather patterns and climate change due to a rising sea level and development in various surrounding areas. In cases where seawater rises impact stormwater outlets, even minor rain events can lead to so-called nuisance floods [34]. DDPG with the tested reward function, regardless of noise added showed the greatest reduction in flooding during these minor rainfall events, situations when nuisance floods would occur. Thus, communities such as Norfolk could benefit greatly from RL solutions. The SWMM tool helps with the design factors, but the ability to learn how to best adjust the valves during rain events could continually improve.

RL application to mitigate flooding in coastal cities is an emerging strategy for this important problem. Using authentic data sets, with the introduction of noise provides a realistic simulation of stormwater systems. DDPG was able to accommodate a complex model considering an integration of rain forecasting, each with imperfect or noisy data. DDPG and simulations can be adapted with different rainfall datasets and models indicative of other cities and as development of infrastructure continues. Thus, one could relatively quickly retrain the algorithm based on the new environment, and hope to see similar results. The application of DDPG, and, to a greater extent reinforcement learning, has application in a wide variety of civil infrastructure systems, not limited to just stormwater infrastructure. Using DDPG and other machine learning strategies to control critical infrastructure has the ability to process multiple, simultaneous factors in near real-time following sufficient training. Furthermore, this paper describes the ability of the machine to accommodate uncertainties that represent real-world factors and can reduce the loss of life and property by dynamically automating stormwater systems.

5. Conclusions

This study demonstrated the viability and success of a DDPG RL implementation in controlling stormwater systems with a reduction of flooding in a variety of storm events observed in Norfolk, VA. Three stormwater control conditions were compared with the passive system: (i) RL using perfect input data, (ii) RL using noisy forecasts, and (iii) RL using noisy state data. We hypothesized that RL would provide better flood mitigation than the passive system by accommodating for noisy forecast or state data that is more representative of the real-world than having perfect knowledge of current and future states. Three DDPG agents were trained by interacting with a SWMM simulation for 10,000 time steps. Following training, the RL policies were tested on a set of previously unseen rainfall events. We found that the DDPG strategies were able to accommodate uncertain data resulting in similar flood mitigation ability between the three data conditions. Compared to the passive system, the three RL implementations reduced flooding by an average of 70.5% on the test events.

The results of this research demonstrate the promise of using RL to control stormwater systems in real-time, even under uncertain conditions. Additional work to further understand how RL performs for stormwater RTC includes using real rainfall forecasts that are longer than the forecasts used here. This would allow control decisions to be made even further in advance of storm events to prepare the system to prevent flooding. Longer forecasts; however, may include more uncertainty and increase the chance of making inappropriate control decisions. Another key topic to advance this research is to explore RL's ability to maintain overall system performance when a sensor or group of sensors is not functioning or is reporting erroneous data. This will be especially critical as both the density of sensor networks and the adoption of stormwater RTC systems continue to increase [12].

Author Contributions: Conceptualization, S.M.S., B.D.B., P.A.B. and J.L.G.; Data curation, S.M.S. and B.D.B.; Formal analysis, S.M.S.; Funding acquisition, J.L.G.; Investigation, S.M.S. and B.D.B.; Methodology, S.M.S., B.D.B. and S.A.; Project administration, S.M.S. and B.D.B.; Resources, S.M.S.; Software, S.M.S.; Supervision, B.D.B. and S.A.; Validation, S.M.S. and B.D.B.; Visualization, S.M.S. and B.D.B.; Writing—original draft, S.M.S.; Writing—review & editing, S.M.S., B.D.B., S.A., P.A.B. and J.L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded as part of two U.S. National Science Foundation grants: Award #1735587 (CRISP-Critical, Resilient Interdependent Infrastructure Systems and Processes) and Award #1737432 (SCC-IRG Track 1: Overcoming Social and Technical Barriers for the Broad Adoption of Smart Stormwater Systems).

Acknowledgments: We gratefully acknowledge the Hampton Roads Sanitation District for access to their high quality rainfall data.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RTC	Real Time Control
RL	Reinforcement Learning
DDPG	Deep Deterministic Policy Gradients
RL	Reinforcement Learning
SWMM	Storm Water Management Model
S	Subcatchment
J	Junction
R	Orifice Valve
MDP	Markov Decision Process
S	State
A	Action

\mathcal{P}	Transition Function
γ	Discount factor
s	Specific State
a	Specific Action
o	Orifice Valve Setting
f	Forecast
D	Depths
E	Expected Depth
f'	Imperfect Forecast

References

1. Ezer, T.; Atkinson, L.P. Accelerated flooding along the US East Coast: On the impact of sea-level rise, tides, storms, the Gulf Stream, and the North Atlantic Oscillations. *Earth's Future* **2014**, *2*, 362–382. [\[CrossRef\]](#)
2. O'Driscoll, M.; Clinton, S.; Jefferson, A.; Manda, A.; McMillan, S. Urbanization effects on watershed hydrology and in-stream processes in the southern United States. *Water* **2010**, *2*, 605–648. [\[CrossRef\]](#)
3. Sweet, W.V.; Park, J. From the extreme to the mean: Acceleration and tipping points of coastal inundation from sea level rise. *Earth's Future* **2014**, *2*, 579–600. [\[CrossRef\]](#)
4. Sörensen, J.; Persson, A.; Sternudd, C.; Aspegren, H.; Nilsson, J.; Nordström, J.; Jönsson, K.; Mottaghi, M.; Becker, P.; Pilesjö, P.; et al. Re-thinking urban flood management—Time for a regime shift. *Water* **2016**, *8*, 332. [\[CrossRef\]](#)
5. Miller, J.D.; Hutchins, M. The impacts of urbanisation and climate change on urban flooding and urban water quality: A review of the evidence concerning the United Kingdom. *J. Hydrol. Reg. Stud.* **2017**, *12*, 345–362. [\[CrossRef\]](#)
6. Smith, A.B.; Katz, R.W. US billion-dollar weather and climate disasters: data sources, trends, accuracy and biases. *Nat. Hazards* **2013**, *67*, 387–410. [\[CrossRef\]](#)
7. Sadler, J.; Goodall, J.; Morsy, M.; Spencer, K. Modeling urban coastal flood severity from crowd-sourced flood reports using Poisson regression and Random Forest. *J. Hydrol.* **2018**, *559*, 43–55. [\[CrossRef\]](#)
8. Sadler, J.M.; Haselden, N.; Mellon, K.; Hackel, A.; Son, V.; Mayfield, J.; Blase, A.; Goodall, J.L. Impact of sea-level rise on roadway flooding in the Hampton Roads region, Virginia. *J. Infrastruct. Syst.* **2017**, *23*, 05017006. [\[CrossRef\]](#)
9. Li, F.; Yan, X.F.; Duan, H.F. Sustainable Design of Urban Stormwater Drainage Systems by implementing detention tank and LID measures for flooding risk control and water quality management. *Water Resour. Manag.* **2019**, *33*, 3271–3288. [\[CrossRef\]](#)
10. Kerkez, B.; Gruden, C.; Lewis, M.; Montestruque, L.; Quigley, M.; Wong, B.; Bedig, A.; Kertesz, R.; Braun, T.; Cadwalader, O.; et al. Smarter Stormwater Systems. *Environ. Sci. Technol.* **2016**, *50*, 7267–7273. [\[CrossRef\]](#)
11. Rathnayake, U.; Anwar, A.F. Dynamic control of urban sewer systems to reduce combined sewer overflows and their adverse impacts. *J. Hydrol.* **2019**, *579*, 124150. [\[CrossRef\]](#)
12. Marchese, D.; Jin, A.; Fox-Lent, C.; Linkov, I. Resilience for Smart Water Systems. *J. Water Resour. Plan. Manag.* **2020**, *146*, 02519002. [\[CrossRef\]](#)
13. Ibrahim, Y.A. Real-Time Control Algorithm for Enhancing Operation of Network of Stormwater Management Facilities. *J. Hydrol. Eng.* **2020**, *25*, 04019065. [\[CrossRef\]](#)
14. Jafari, F.; Mousavi, S.J.; Yazdi, J.; Kim, J.H. Real-time operation of pumping systems for urban flood mitigation: Single-period vs. multi-period optimization. *Water Resour. Manag.* **2018**, *32*, 4643–4660. [\[CrossRef\]](#)
15. Mounce, S.; Shepherd, W.; Ostojin, S.; Abdel-Aal, M.; Schellart, A.; Shucksmith, J.; Tait, S. Optimisation of a fuzzy logic-based local real-time control system for mitigation of sewer flooding using genetic algorithms. *J. Hydroinform.* **2020**, *22*, 281–295. [\[CrossRef\]](#)
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning, a Bradford Book*; MIT Press: Cambridge, MA, USA, 1998; Volume 2015, p. 2.

17. Mullapudi, A.; Lewis, M.J.; Gruden, C.L.; Kerkez, B. Deep reinforcement learning for the real time control of stormwater systems. *Adv. Water Resour.* **2020**, *140*, 103600. [\[CrossRef\]](#)
18. Wang, C.; Bowes, B.; Tavakoli, A.; Adams, S.; Goodall, J.; Beling, P. Smart Stormwater Control Systems: A Reinforcement Learning Approach. In Proceedings of the ISCRAM Conference Proceedings—17th International Conference on Information Systems for Crisis Response and Management, Blacksburg, VA, USA, 24–27 May 2020; Hughes, A.L., McNeill, F., Zobel, C., Eds.; pp. 2–13.
19. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
20. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [\[CrossRef\]](#)
21. Bowes, B.D.; Tavakoli, A.; Wang, C.; Heydarian, A.; Behl, M.; Beling, P.A.; Goodall, J.L. Flood mitigation in coastal urban catchments using real-time stormwater infrastructure control and reinforcement learning. *J. Hydroinform.* **2020**, jh2020080. [\[CrossRef\]](#)
22. Rayner, S.; Lach, D.; Ingram, H. Weather forecasts are for wimps: why water resource managers do not use climate forecasts. *Clim. Chang.* **2005**, *69*, 197–227. [\[CrossRef\]](#)
23. Jan van Andel, S.; Price, R.K.; Lobbrecht, A.H.; van Kruiningen, F.; Mureau, R. Ensemble precipitation and water-level forecasts for anticipatory water-system control. *J. Hydrometeorol.* **2008**, *9*, 776–788. [\[CrossRef\]](#)
24. Duan, H.F.; Li, F.; Tao, T. Multi-objective optimal design of detention tanks in the urban stormwater drainage system: uncertainty and sensitivity analysis. *Water Resour. Manag.* **2016**, *30*, 2213–2226. [\[CrossRef\]](#)
25. Hartono, P.; Hashimoto, S. Learning from imperfect data. *Appl. Soft Comput.* **2007**, *7*, 353–363. [\[CrossRef\]](#)
26. Gironás, J.; Roesner, L.A.; Rossman, L.A.; Davis, J. A new applications manual for the Storm Water Management Model (SWMM). *Environ. Model. Softw.* **2010**, *25*, 813–814. [\[CrossRef\]](#)
27. Jang, S.; Cho, M.; Yoon, J.; Yoon, Y.; Kim, S.; Kim, G.; Kim, L.; Aksoy, H. Using SWMM as a tool for hydrologic impact assessment. *Desalination* **2007**, *212*, 344–356. [\[CrossRef\]](#)
28. Gu, S.; Lillicrap, T.; Sutskever, I.; Levine, S. Continuous deep q-learning with model-based acceleration. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 2829–2838.
29. Choo, B.; Crannel, G.; Adams, S.; Dadgostari, F.; Beling, P.A.; Bolcavage, A.; McIntyre, R. Reinforcement learning from simulated environments: An encoder decoder framework. In Proceedings of the Spring Simulation Conference (SpringSim), Fairfax, VA, USA, 18–21 May 2020; pp. 1–12.
30. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.
31. Plappert, M. keras-rl. 2016. Github Repository. Available online: <https://github.com/keras-rl/keras-rl> (accessed on 15 August 2019).
32. McDonnell, B.E.; Ratliff, K.; Tryby, M.E.; Wu, J.J.X.; Mullapudi, A. PySWMM: The Python Interface to Stormwater Management Model (SWMM). *J. Open Source Softw.* **2020**, *5*, 2292. [\[CrossRef\]](#)
33. Hu, Q.S.; Skaggs, K. Accuracy of 6–10 day precipitation forecasts and its improvement in the past six years. In Proceedings of the 7th NOAA Annual Climate Prediction Application Science Workshop, Norman, OK, USA, 24–27 October 2009.
34. Sheridan, S.C.; Pirhalla, D.E.; Lee, C.C.; Ransibrahmanakul, V. Atmospheric drivers of sea-level fluctuations and nuisance floods along the mid-Atlantic coast of the USA. *Reg. Environ. Chang.* **2017**, *17*, 1853–1861. [\[CrossRef\]](#)

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).