Ellicott City Disasters III

*Building a Real-Time Statistical Flood Model for Improving Early Warning Systems in Ellicott City, Maryland*

 **Technical Report**

Final Draft – August 6th, 2020

Erika Munshi (Project Lead)

Ryan Hammock

Eli Orland

Alina Schulz

***Advisors***

Dr. John Bolten, NASA Goddard Space Flight Center (Science Advisor)

Dr. Sujay Kumar, NASA Goddard Space Flight Center (Science Advisor)

Perry Oddo, Science Systems and Applications, Inc. (Science Advisor)

Brian Cleary, Howard County Storm Water Management Division (Science Advisor)

Previous Contributors:

Terra Edenhart-Pepe

Julio Peredo

Caroline Resor

Callum Wayman

Scott Cunningham

Jonathan Donesky

Matthew Pruett

# 1. Abstract

As flood events in the United States grow in frequency and intensity, the uses of applied remote sensing analyses are increasingly necessary for effective flood monitoring and warning systems. The NASA DEVELOP Ellicott City Disasters III project investigated the use of machine learning for applications in flood risk detection to support the improvement of early warning systems in Ellicott City, Maryland. To strengthen the efforts of the Howard County Office of Emergency Management (OEM) in building a more robust flood monitoring system, this term built on the predictive capability of the long-short term memory (LSTM) model created by the second term of this DEVELOP project to create a Sequentially Trained Real-time EstimAted Model (STREAM). Enhancements to the model included the integration of both real-time and predicted weather products from the National Weather Service to increase predictive capacity. These weather products were supplemented by stream gauge data from the OEM as well as real-time radar products. The resultant flood risk model was trained to evaluate input variables and predict stage height in Ellicott City in real time. The model, upgraded to predict stage height up to 8 hours in advance, was incorporated into an online dashboard in a user-friendly interface. The project demonstrated the potential for integration of open data and NASA Earth observations into a flood risk forecasting tool capable of informing real-time decision-making.

**Key Terms**

flooding, extreme events, deep learning, machine learning, runoff

# 2. Introduction

* 1. ***Background Information***

Flooding is one of the most destructive natural disasters in the United States, costing billions of dollars in damages annually and endangering thousands of lives (Thakali et al., 2017). Because flood frequency and severity are expected to increase due to climate variability, it is of great importance to developing mechanisms to reduce vulnerability and improve resilience (Ganguly, Nahar, & Hossain, 2018; Hirabayashi et al., 2013). Flash floods, in particular, pose a unique threat to infrastructure and livelihoods due to their rapid onset, and increasing the warning time prior to a flood event can significantly reduce negative outcomes (Alipour et al., 2020).

The Mid-Atlantic region of the United States is prone to flooding (Smith & Smith, 2015). Ellicott City, Maryland (*Figure 1)* is particularly at risk because it lies in a steep river valley composed of a layer of low-drainage soil overlying low-permeability granite bedrock (United States Department of Agriculture Natural Resources Conservation Service, 2008). A contiguous United States-wide analysis that quantified relative flash flood severity on a scale of zero to one (where “flashiness” is the difference between the peak discharge and action stage discharge normalized by the flooding rise time and basin area) gave Ellicott City a maximum score of 1.0 (Saharia et al., 2017). The historic downtown is specifically situated at the junction of several tributaries and the Patapsco River, making it uniquely vulnerable to flash flooding. According to the National Weather Service, the city has been struck with two 1,000-year rainfall events in recent years: one in July 2016 and another in May 2018 (Halverson, 2018). Frequency and intensity of flooding can be caused by a confluence of factors, including extreme weather and urbanization, with the latter leading to an increase in impervious surface area (Viterbo et al., 2020). In addition to these issues, Ellicott City has the added stress from an aging drainage system, designed to manage 100-year floods, but the recent 1,000-year rainfall events have overwhelmed the city’s current infrastructure.

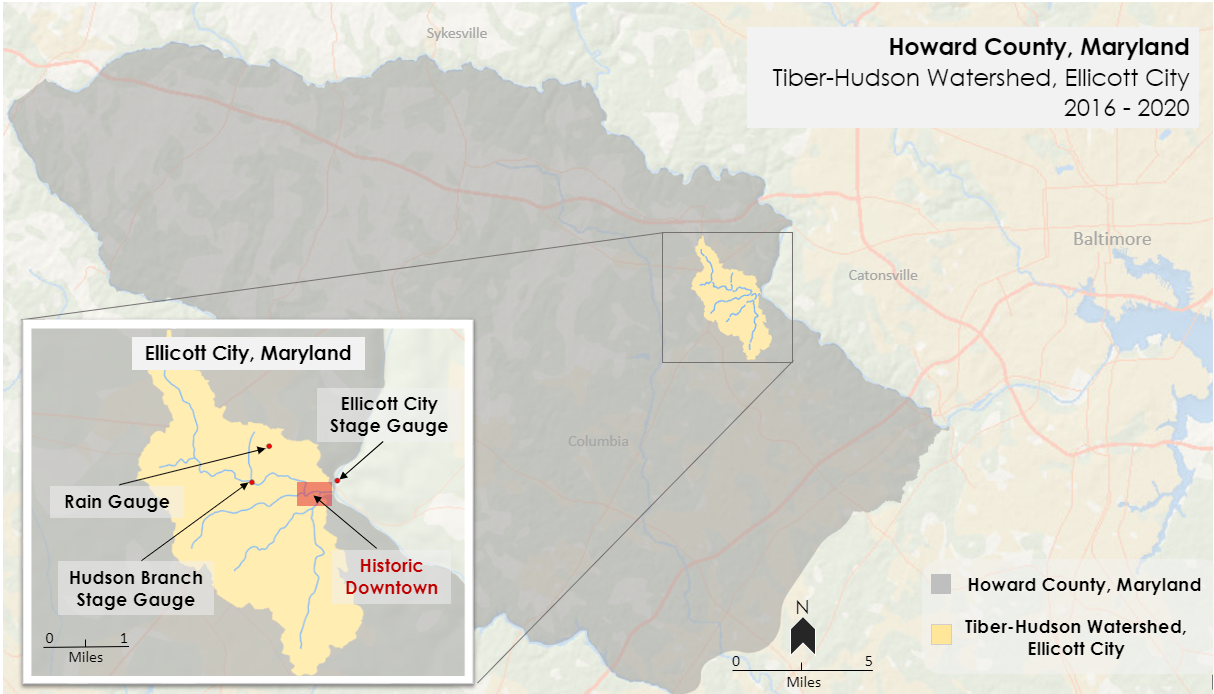


Figure 1. Project study area of Ellicott City, relative to Howard County in Maryland, featuring rain and stage gauges located in and near the Tiber-Hudson watershed.

The use of machine learning applications for flood and risk predictions is a field of active research. Neural networks, a subset of machine learning models, are frequently used to analyze floods (Mosavi, Ozturk, & Chau, 2018). These models attempt to iteratively explore the importance of various input parameters on flood severity. Neural networks require training data to perform their analyses, which can be provided from ground-based and remote sensing data. One particularly well-performing recurrent neural network used for modeling complex, non-linear runoff situations is known as the Long Short-Term Memory (LSTM) network (Xiang, Yan, & Demir, 2020). The advantage of the LSTM is its ability to learn long-term dependencies between variables provided as inputs and outputs of the network. This type of memory is particularly useful for modeling relationships between variables that do not have simple linear relationships, such as precipitation rates and soil moisture (Kratzert, Klotz, Brenner, Schulz, & Herrnegger, 2018).

The previous terms of this DEVELOP project created different flood severity models based on historical data to test the feasibility of applying machine learning to flood prediction. The first term’s model, Flood Learning Model Environment (FLuME), used a statistical flood risk approach which confirmed that machine learning provides a feasible way to integrate NASA Earth observations into public safety and emergency protocols. The second term’s model, Flood Learning Environment and Severity Hub (FLASH), was built from a deep learning LSTM architecture and used historical data from 2011 to 2020. Using this approach, FLASH identified the parameters most influential in predicting stage height of severe events and showed reasonable confidence when tested via a hindcast experiment based on *in situ* data from the 2018 flood event.

This project expanded upon the FLASH model by incorporating near real-time data to expand the predictive capability of the model and provide predictions up to eight hours in advance. This upgraded model, called STREAM (Sequentially Trained Real-time EstimAted Model), leverages the LSTM modeling approach in tandem with real-time data provided from NASA Earth observations and project partners to create real-time forecasted stage height output available to the Howard County Office of Emergency Management (OEM) through an online weather-monitoring platform called OneRain. Once visible on OneRain, STREAM can be used to aid real-time decision-making regarding evacuations and other emergency management procedures.

* 1. ***Project Partners & Objectives***

This project was completed with support from the Howard County Government OEM, the Howard County Government Stormwater Management Division (SMD), and the National Oceanic and Atmospheric Administration (NOAA) National Weather Service (NWS) Baltimore-Washington Weather Forecast Office. The Howard County OEM is responsible for managing and overseeing emergency preparedness, particularly for flood management. Our partners at the OEM, SMD, and NWS Baltimore-Washington Weather Forecast Office were interested in using remote sensing to facilitate proactive decision-making. To support this effort, the DEVELOP team focused on enhancing a statistical model to predict flood severity in Ellicott City. Building on the work from previous DEVELOP terms, the team enhanced a machine learning model, combining NASA Earth observations, stage height data, and forecasted precipitation to predict flood severity in near real-time. In accordance with the project objectives, the team modified the statistical flood severity model from the previous term (FLASH) to build on the LSTM network and incorporate real-time observations. To provide an interactive tool for our partners, the team then created a series of scripts to run the model iteratively, producing real-time forecasts on OneRain.

# 3. Methodology

***3.1 Data Acquisition***

The team used a variety of data sources including *in situ* gauge data and NOAA and NASA modeled weather products to train and test the machine learning model. Discharge and stage height on the Patapsco River and Hudson branch were provided via *in situ* gauges operated by the Howard County OEM and the United States Geological Survey (USGS). Precipitation forecasts were provided by the High-Resolution Rapid Refresh (HRRR) model generated using NASA/NOAA Geostationary Operational Environmental Satellite-16 (GOES-16) Earth observations.

***3.1.1 In situ Gauges***

The Howard County OEM and the USGS operate several *in situ* gauges at and around Ellicott City that provide data of high temporal resolution which proved vital to our research (Table 1). Howard County operates three gauges within the bounds of the study area: one measuring stage height and discharge of the Patapsco River; one measuring the discharge of the Tiber, a stream that runs through the center of the watershed; and a rain gauge within the city. These data were obtained using the Howard County OneRain portal. The USGS operates two stream gauges on the Patapsco River, one upstream from the city at Hollofield, and one downstream at Catonsville. These values were obtained from the National USGS Water Data Portal. *Figure 2* shows the locations of all of the gauges.

Table 1

*In situ datasets and parameters used to train the model and evaluate its performance.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Provider** | **Source** | **Temporal Resolution** | **Use** |
| Stage Height, Discharge | USGS | USGS Surface Water Database | 5 min | Model Training & Evaluation |
| Stage Height, Discharge, Precipitation | Howard County OEM | OneRain Portal, Howard County OEM | 15 min | Model Training & Evaluation |

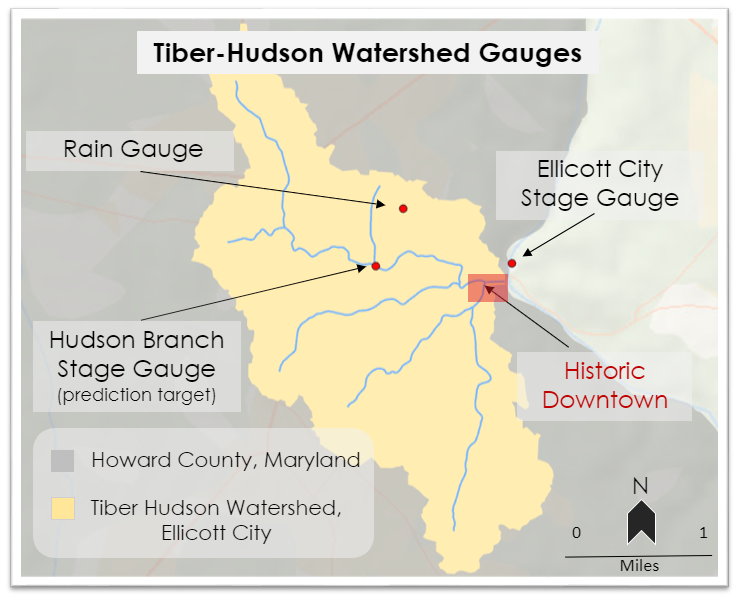


Figure 2. A map showing the gauges and tributaries in the Tiber-Hudson watershed of Ellicott City, Maryland.

***3.1.2 Real-Time and Forecast Precipitation Datasets***

The team acquired real-time forecasted precipitation data utilizing NOAA’s High-Resolution Rapid Refresh Model (HRRR) provided through the University Corporation for Atmospheric Research’s (UCAR) Unidata THREDDS Data Server (TDS). The products available through the HRRR model provide forecasted precipitation (Table 2). A model initialization starts every hour and provides up to thirty-six hours of forecasted output in fifteen-minute frames. This spatial dataset serves as a “future radar” and is used in STREAM to provide future precipitation values.

Table 2

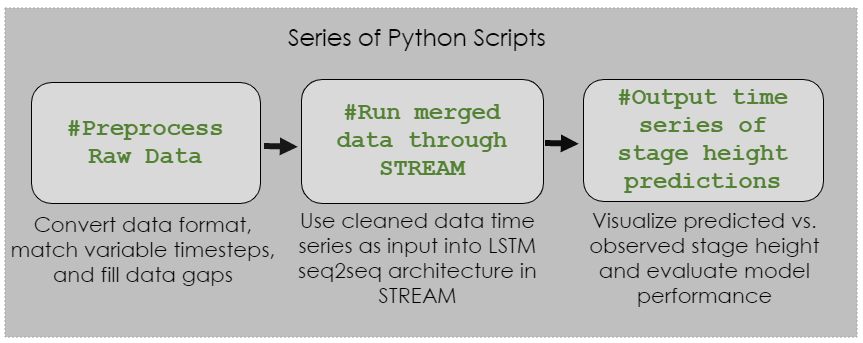
This table lists the dataset that is used to provide real-time precipitation and forecast information for the flood risk model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dataset** | **Level** | **Provider(s)** | **Parameter(s)** | **Temporal Resolution** | **Use** |
| High-Resolution Rapid Refresh Model | N/A | NOAA/UCAR | Forecast Radar Reflectivity, Forecast Precipitation | Available in 15 min frames, refreshed hourly | Real-time Data Input |

***3.2 Data Processing***

*3.2.1 STREAM (Sequentially Trained Real-time EstimAted Model)*

Several Python scripts developed in previous iterations of the statistical flood risk model were replicated or modified to process new data for STREAM (*Figure 3*). A script was used to collect precipitation, soil moisture, and discharge data and merge columns based on shared timestamps. The script also checked for and filled missing values. Through the duration of the term, the team worked with this dataset, but ultimately utilized only two features from this historical dataset: the stage height of the Hudson Branch and the predicted HRRR precipitation values at the gauge site. This choice was the result of needing to train the model with the same expected inputs that are available during real-time implementation.



*Figure 3.* This series of Python scripts simplify the model development process to three main stages: preprocessing raw data, merging data, and outputting a time series of stage height predictions.

The tabular dataset of the Hudson Branch stage height and recorded precipitation was used for a machine learning regression problem known as supervised learning, where a specific algorithm is tasked with finding a functional relationship between sets of input and output variables. To replicate the real-time forecast STREAM receives on the OneRain portal, we extracted the hourly recorded rainfall during each eight-hour stream forecasting period and assigned this series of values as the forecasted rainfall for the prediction interval. In doing so, we constructed a supervised learning problem such that the model receives both stream gauge information prior to the forecasting period and the exact recorded rainfall during the forecasting period in order to learn the functional relationship between precipitation and river response. This data structure is furthermore representative of the streamflow and real-time precipitation forecast information the model receives to make real-time streamflow measures when hosted on OneRain.

For the LSTM-based model, we further required the combined two-dimensional dataset (*observation*, *features observed*) to be converted to a three-dimensional dataset of form (*samples, number of observations [timesteps] per sample, features per timestep*). As such, we transformed our combined dataset so that each sample contained the last *n* timesteps of all observations (designated as our “*features”*); these input features are matched to the following *m* future timesteps (designated as our *targets*). This allowed a machine-learning algorithm to learn the implicit relationships between the information encoded in the prior *n* timesteps and the response of the target variable(s). Within a real-time framework, this training process provided the means to forecast river response for the next *m* timesteps based on the information recorded in the past *n* timesteps.

The model inputs consist of an array of values representing each input variable at each timestep between the present time and the last 16 timesteps in the past. In *Figure 4*, this is represented by all the data contained within the bounded box labeled Input Data. The output paired to each window of inputs is a single value at a user-specified timestep in the future, represented in *Figure 4* by the points labeled Predicted Values.

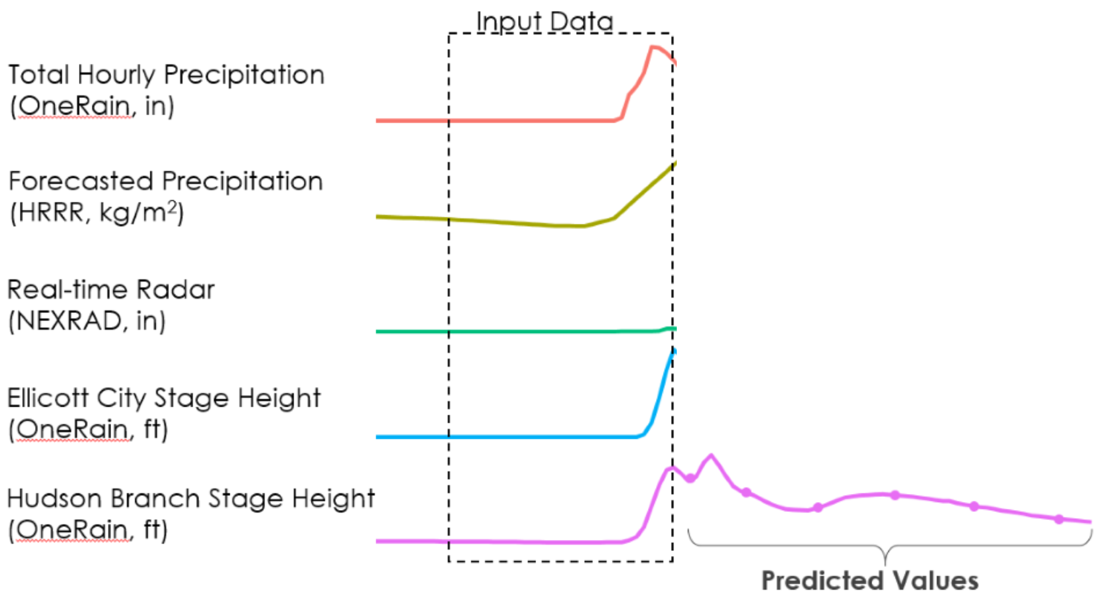
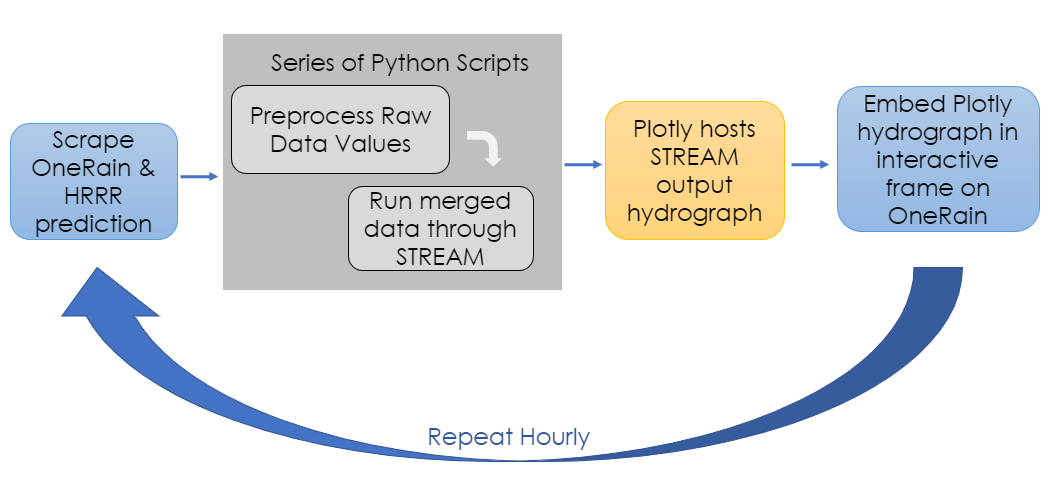


Figure 4. Time series of input variables to STREAM with example box showing input data and example predicted data.

In order to build a robust machine learning model, we provided training data from which our model learned the implicit hydrologic relationships encoded in our time series, as well as a set of unseen data used to test the model’s general performance capacity. The model learns the relationship between inputs and outputs in the training subset and attempts to predict outputs given inputs in its testing data subset. The team divided our training and testing datasets such that the years 2016 and 2017 were used for training while 2018 and 2019 were used for testing. This partitioning was chosen so that each subset would contain one of the two major floods, which occurred in 2016 and 2018.

*3.2.2 Howard County OneRain Dashboard*

In order to make STREAM forecasts visible in real-time, a series of scripts were developed to acquire real-time data. These data were fed through the trained model, and an output image was embedded in the OneRain dashboard developed by the team. The following methodology describes the scripts used in this process (*Figure 5*).



*Figure 5.* This flowchart represents the steps taken to produce real-time updates to the OneRain dashboard featuring the hydrograph with predicted stage height values.

This workflow, which repeats and refreshes outputs hourly, begins with data acquisition from the OneRain portal and the HRRR prediction repository. A series of Python scripts preprocess the raw data values using interpolation, gap filling, and normalization techniques. They also merge the new data on common hourly timesteps and run the newly acquired data through STREAM. The model’s outputs are pushed to the online data visualization platform, Plotly, to generate an iteratively updated hydrograph. Finally, the Plotly hydrograph is embedded in an interactive frame hosted on the OneRain Dashboard created by the team this term. A sample output can be seen in *Figure 6*.

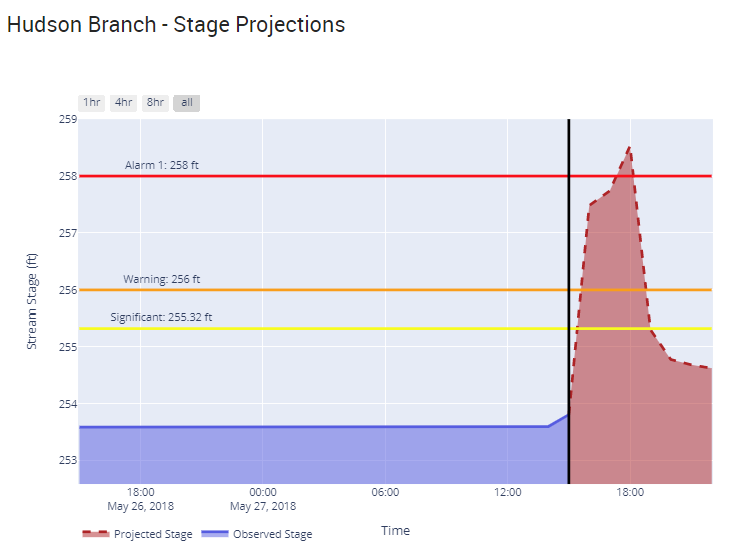


Figure 6. Snapshot of the interactive stream stage height prediction widget embedded in the OneRain Dashboard capable of showing model forecasts 1, 4, and 8 hours from the current time.

***3.3 Data Analysis***

The LSTM model developed during this term is a type of recurrent neural network (RNN) used in machine learning and is capable of processing sequential or temporal information (Hochreiter & Schmidhuber, 1997). The LSTM improves upon an RNN cell by allowing for the explicit incorporation of long-term sequential dependencies through its internal state, which is iteratively updated between timesteps. This internal state acts as a matrix of latent information from previous timesteps upon which relevant information for predictions at the current timestep can be drawn (*Figure 7*).

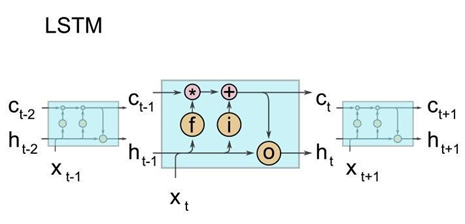


Figure 7. Illustration of an LSTM cell, where **f** stands for the forget gate, **i** for the input gate, and **o** for the output gate. **c**t denotes the cell state at time step t and **h**t is the hidden state. Reprinted from Kratzert et al. (2018).

As neural networks have the capability to approximate any continuous function (Hornik et al., 1989), they are suitable for modeling nonlinear physical relationships. For example, Abrahart & See (2007) demonstrate the ability of neural networks to approximate common nonlinear hydrologic processes; furthermore, Kratzert et al. (2018) demonstrate the efficacy of LSTM cells for rainfall runoff modeling in particular—a primary objective for our study.

While FLASH excelled at predicting single values into the future, the requirement to generate multiple outputs drove the decision to move towards a different approach. Resultantly, a major enhancement to the previously developed LSTM model during this term featured the sequence-to-sequence (seq2seq) encoder/decoder architecture (Sutskever et al., 2014) with a global Luong attention mechanism (Luong et al., 2015). The benefit here lies in its ability to create predictions of different sequence lengths to the input data*.* For real-time applications, this means having the ability to use the past *n* hours to look 1, 6, and 12- hours into the future using a single model run.

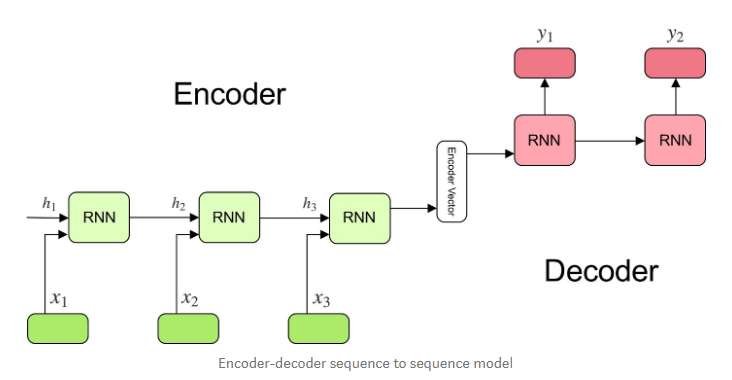


Figure 8. Sequence-to-sequence (seq2seq) encoder/ decoder diagram illustrating the model's enhanced ability to create outputs of different sequence lengths than its inputs. Reprinted from [Kostadinov](https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346) (2019).

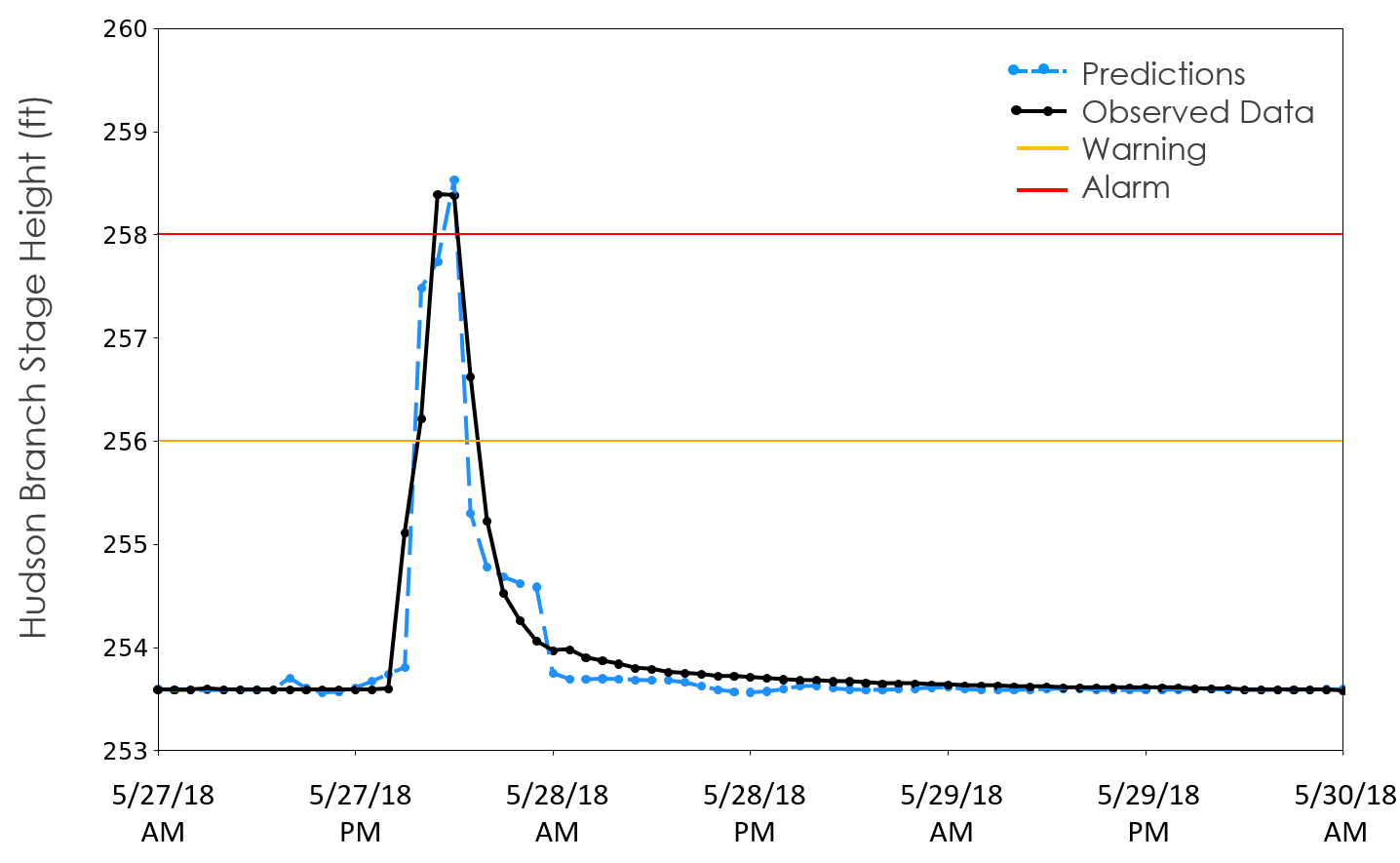
*Figure 8* demonstrates a simplified encoder-decoder architecture. Here, an RNN cell—in our case, an LSTM cell—receives an input sequence of arbitrary length and stores a representation or “understanding” of this sequence, such as the relevant stream gauge or high precipitation values encoded in a time series. This cell is referred to as the *encoder*, with a representation of the last timestep referred to as the *hidden state*, per *Figure 7.* The memory of the sequence as a whole is stored in the encoder’s *cell state.* Both the hidden and cell states (combined, referred to as the *encoder vector*) are passed on to a second RNN cell, referred to as the *decoder*. The decoder receives both states and is trained to create predictions representing an output sequence of arbitrary length at single timestep intervals. For applications to stream gauge predictions, this would include iterative predictions of stream gauge measurements at fixed timestep intervals. In summary, the encoder works to learn the relevant information within an input sequence, which is then passed on the decoder. The decoder then uses this information to generate meaningful output. Finally, the global attention mechanism makes the hidden state from each timestep available to the decoder. This provides information that is still relevant for the decoder but was not captured within the encoder’s cell state. To script our model, we relied on the Keras frontend API and a TensorFlowbackend. We optimized our model with the Adam optimizer (Kingma & Ba, 2014), and train using a mean squared error (MSE) loss function.

# 4. Results & Discussion

***4.1 Model Performance***

With reference to the work from previous terms, the switch from a many-to-one to a many-to-many (seq2seq) approach provides STREAM with the capability to forecast sequences of river activity, as opposed to predicting a fixed time step (15 min, 1 hr, 3 hr) into the future. Furthermore, STREAM is capable of running on a GPU backend, significantly reducing training time. Training takes between 2-10 minutes using a single GPU processor run on Google’s Colaboratory scripting environment.

During the current term of the project, STREAM was trained on data from 2016 to 2017 and tested on data from 2018 to 2019. It receives the last 16 hourly timesteps of river stage height data and the most recent HRRR forecast for Ellicott City. STREAM then predicts the next 8 hr of river activity. Our 8 hr forecasts incur a mean absolute error of 0.07 ft across our testing dataset and an NSE of 0.44. For the hours of the 2018 flood (May 27th, 3 pm-8 pm ET), the absolute model error is 0.85 ft with an NSE of 0.48. For the flood period including the extended time before and after the event, model error is 0.13 ft with an NSE of 0.89 (*Figure 9*). While an NSE value of 0.44 for the explicit flood hours demonstrates moderate predictive power, STREAM nonetheless correctly anticipates the approximate magnitude and timing of the 2018 flood. This demonstrates STREAM’s ability to approximate the highly non-linear physical relationships that govern the surface-runoff relationships in the greater Ellicott City area such that the community would have likely had up to 3-4 hours of advance warning of an imminent flood, provided the correct forecast information via the HRRR model.

*Figure 9*. Sample results from STREAM showing the forecasted response to the 2018 Ellicott City flood in 8-hour prediction intervals. Data inputs include the past 16 hours of Hudson Branch stage height observations and 8-hour HRRR predicted rainfall accumulation.

With the likelihood of future flood events, it is important to put the capabilities provided by STREAM in the context of the current availability of resources in Howard County. Residents have access to video and photo feeds documenting the current stage height of local streams, in addition to weather forecasts for the area and quantitative measurements of the river stage height on the OneRain Portal. However, the current integration of STREAM onto OneRain represents the first time Howard County residents will have access to a predictive model that combines real-time river stage height and forecasted weather information, thus joining the previously available resources into a single tool to anticipate flooding hazards. As a result, STREAM provides the capability to anticipate river stage height explicitly, doing so several hours in advance in order to provide critical warnings to residents who may be impacted by local flooding. While no model is perfect, we are encouraged by the early results provided by STREAM, and look forward to further improvements that will continue to provide timely warnings to the residents of Ellicott City.

***4.2 Future Work***

While STREAM is an improvement from the previous terms’ models and operates as a near real-time flood prediction tool, there are additional datasets that can be incorporated into STREAM to improve longer-term stage height prediction up to 12 hours in advance. Among those additional datasets includes the integration of real-time 1-hour quantitative precipitation estimation (QPE) products from the Sterling, VA (LWX) NEXRAD (Next-Generation Radar) radar site. QPE products from NEXRAD provides precipitation values beyond a single point and provides a spatial distribution of precipitation values across the entire study area while serving as a supplement to *in situ* rain gauges. The implementation of a NEXRAD precipitation product for STREAM requires both real-time and archived data to train the model. During this term, the team explored methods of obtaining real-time and achieved NEXRAD Level III one-hour precipitation (N1P) product from UCAR’s Unidata THREADDS Radar Server. The product has yet to be implemented into the model due to the absence of consistent archive data to train STREAM. Future work could focus on obtaining or collecting the NEXRAD N1P data over an extended time period to properly train and implement the product into the model.

STREAM would additionally benefit from the integration of a historical archive of the HRRR model’s precipitation predictions since it makes its real-time predictions using HRRR as an input. During this term, the team found a suitable archive being kept at the University of Utah on the Center for High-Performance Computing (CPHC) Pando Archive System which covers our study period (2016- 2020). There are a series of scripts developed by the archive manager, Brian Blaylock, on a GitHub repository (Blaylock et al., 2017). The team followed the tutorials to gather the total precipitation accumulation variable at hourly intervals with 8-hour forecasts using Python scripts in the Google Colaboratory environment. However, this variable has yet to be incorporated into STREAM due to data format limitations. Future work could focus on creating a workflow for extracting the Ellicott City pixel from the continental US scene that these scripts currently produce. See the Appendix for a list of data sources with integration potential into future iterations of STREAM.

# In addition to building in auxiliary data sources, continuing work on the Ellicott City project should pursue additional analyses in validating and calibrating model performance to the Tiber-Hudson watershed. This could include integrating the additional gauges installed by the Howard County OEM throughout 2020. Another approach could integrate the two-dimensional model produced using a convolutional LSTM approach into the OneRain dashboard as an additional independent model displaying forecasts. Lastly, using computer vision algorithms to recognize water level from the live-feed cameras currently used only as visual confirmation would expand the model’s real-time data sources. Fortunately for the continuation of this 3-term project, oversight for model performance and development will be conducted by Goddard Space Flight Center/SSAI.

# 5. Conclusions

The team’s results yielded a near real-time hourly updating machine learning model called STREAM, which predicts stage height in a stream within the flash flood-prone Tiber-Hudson watershed in Ellicott City. Using the progress made during the first two terms of this project, STREAM uses an LSTM architecture enhanced with a sequence-to-sequence encoder/decoder and a global attention mechanism written using Python’s TensorFlow open-access libraries. The model harnessed the power of the two inputs, HRRR model precipitation predictions and Hudson Branch stage height, to predict stage height at hourly intervals up to 8 hours into the future.

The Howard County Office of Emergency Management, focused on increasing public safety and effective emergency management response, benefits from the model’s integration into their existing gauge monitoring portal, OneRain. This new dashboard, called “NASA Forecasts,” outputs every 30 minutes to display STREAM’s stage height predictions, along with real-time local radar, observed precipitation and stream gauge activity in the watershed, and OEM’s Twitter feed. Consolidating all of the observations and predictions that focus on flash flood preparedness in one accessible location has helped build the partner’s ability to make informed decisions about responding to public safety threats. Lastly, prior to this novel machine learning approach to flash flood prediction with limited data sources, the partners had no early warning system. While significant infrastructure reinforcement and change initiatives have been pursued since the last severe event in 2018, the addition of STREAM’s predictions gives OEM a unique and powerful new tool to enhance public safety in Ellicott City.

# 6. Acknowledgments

The team would like to thank everyone that contributed to the progress made over the course of this term. We are grateful for the support we received from our Science Advisors at Goddard Space Flight Center: Dr. John Bolten, Dr. Sujay Kumar, and Perry Oddo. We would like to thank our DEVELOP Lead/Fellow, Darcy Gray, for her continued support, guidance, and advice throughout the term. We are also grateful for the contributions made by the previous Ellicott City Disasters teams. Lastly, we would like to recognize our partners at the Howard County Office of Emergency Management, the Howard County Stormwater Management Division, and the NOAA-NWS Baltimore-Washington Forecast Office.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

This material is based upon work supported by NASA through contract NNL16AA05C.

# 

# 7. Glossary

**1,000-year rainfall event** – hydrologic event that has a 0.1 percent chance of happening in a given year based on its extreme magnitude

**Earth observations** – satellites and sensors that collect information about the Earth’s physical, chemical, and biological systems over space and time

**Flashiness** – experiences of rapid increase in flow shortly after the onset of a precipitation event, and an equally rapid return to base conditions shortly after the end of the precipitation event

**Google Earth Engine API** – cloud-based platform for geospatial analysis

**Hidden layer** – a layer between model input and output layers where the model has generated an equation that assigns weights for each input to approximate the output

**Long Short-Term Memory (LSTM)** – an artificial recurring neural network used in machine learning that allows for memory retention in a learning environment

**Machine learning** – a branch of artificial intelligence in which a computer system employs algorithms and statistical models to effectively extract meaning inferred from patterns related to a specific task or inquiry; this process is typically iterative and involves a learning component by which inputs are acted upon and outputs are fed back through statistical models to improve outputs

**Nash-Sutcliffe Efficiency (NSE) coefficient** – a common metric used to evaluate the predictive power of hydrological models, expressed as a value between negative infinity and 1.0 where 1.0 corresponds to a perfect match between modeled and observed values

**Neural network** – computing systems inspired by animal brains that “learn” by completing example tasks using artificial neurons called “nodes”.

**Sequence-to-sequence structure** – deep learning method that pairs with an LSTM to enhance forecasting by considering long term dependencies and multiple outputs

**Stage height** – level (measured by height, often in feet) of water in a channel

**Statistical model** – a mathematical model that embodies a set of assumptions concerning data generation and can be used to make predictions

**Stream discharge** – the rate at which water passes a specified point, measured in volume per second

**Tensor** – a multi-dimensional array which supports the machine learning training process

**United States Geological Survey (USGS)** – the federal agency responsible for collecting data from and maintaining the river gauges used in this study

# 8. References

Abrahart, R. J., & See, L. M. (2007). Neural network modeling of non-linear hydrological relationships. *Hydrology and Earth System Sciences*, *11*(5), 1563–1579. <https://doi.org/10.5194/hess-11-1563-2007>

Alipour, A., Ahmadalipour, A., Abbaszadeh, P., & Moradkhani, H. (2020). Leveraging machine learning for predicting flash flood damage in the Southeast US. *Environmental Research Letters, 15.* <https://doi.org/10.1088/1748-9326/ab6edd>

Blaylock B., J. Horel, and S. Liston, (2017). Cloud archiving and data mining of High-Resolution Rapid Refresh model output. *Computers and Geosciences*. 109, 43-50. https://doi.org/[10.1016/j.cageo.2017.08.005](https://doi.org/10.1016/j.cageo.2017.08.005)

Ganguly, K., Nahar, N., & Hossain, B. (2018). A machine learning-based prediction and analysis of flood affected households: A case study of floods in Bangladesh. *International Journal of Disaster Risk Reduction, 34,* 283-294. https://doi.org/10.1016/j.ijdrr.2018.12.002

GOES-16 Series Program, (2019). NOAA GOES-R Series Advanced Baseline Imager (ABI) Level 0 Data. NOAA National Centers for Environmental Information. Doi:10.25921/tvws-w071.

Halverson, J. B. (2018). “Flood City, USA”: The sorrowful tale of Ellicott City, Maryland. *Weatherwise*, *71*(5), 48. Retrieved from https://search.proquest.com/docview/2159933219?accountid=28155

Hirabayashi, Y., Mahendran, R., Koirala, S., Konoshima, L., Yamazaki, D., Watanabe, S., … Kanae, S. (2013). Global flood risk under climate change. *Nature Climate Change*, *3*(9), 816-821. <https://doi.org/10.1038/nclimate1911>

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. <https://doi.org/10.1162/neco.1997.9.8.1735>

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. <https://doi.org/10.1016/0893-6080(89)90020-8>

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization, 1–15. Retrieved from <http://arxiv.org/abs/1412.6980>

Kostadinov, S. (2019). Understanding encoder-decoder sequence to sequence models. *Towards Data Science.* https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall-runoff modelling using Long Short-Term Memory (LSTM) networks. *Hydrology and Earth System Sciences, 22*(11). <https://doi.org/10.5194/hess-22-6005-2018>

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing,* 1412-1421. <https://doi.org/10.18653/v1/d15-1166>

Mosavi, A., Ozturk, P., & Chau, K. W. (2018). Flood prediction using machine learning models: Literature review. *Water*, *10*(11), 40. https://doi.org/10.3390/w10111536

Saharia, M., Kirstetter, P.-E., Vergara, H., Gourley, J. J., Hong, Y., & Giroud, M. (2017). Mapping flash flood severity in the United States. *Journal of Hydrometeorology*, *18*(2), 397–411.

Smith, B. K., & Smith, J. A. (2015). The flashiest watersheds in the contiguous United States. *Journal of* *Hydrometeorology, 16*, 2365-2381. https://doi.org/10.1175/JHM-D-14-0217.1

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

Thakali, R., Bhandari, R., Kandissounon, G. A., Kalra, A., & Ahmad, S. (2017). Flood risk assessment using the updated FEMA floodplain standard in Ellicott City, Maryland, United States. *Proceedings of the World Environmental and Water Resources Congress 2017*, (pp. 280-291). Sacramento, California: American Society of Civil Engineers. Retrieved from <https://digitalscholarship.unlv.edu/fac_articles/449>

Viterbo, F., Mahoney, K., Read, L., Salas, F., Bates, B., Elliott, J., … Cifelli, R. (2020). A multiscale, hydrometeorological forecast evaluation of National Water Model forecasts of the May 2018 Ellicott City, Maryland, Flood. *Journal of Hydrometeorology, 21, 475*–499.

Xiang, Z., Yan, J., Demir, I. (2020). A rainfall-runoff model with LSTM-based sequence-to-sequence learning. *Water Resources Research, 56*(1),910-922*.* <https://doi.org/10.1029/2019WR025326>

# 9. Appendix

List of Data Sources with Potential for STREAM Integration

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Years Available** | **Source** | **Notes on Integration** |
| Forecasted Reflectivity | 2011 - 2020 | NOAA/ NEXRAD  (accessed via UCAR) | High real-time integration potential; requires continued data acquisition scripting & data preprocessing |
| Archived HRRR Precipitation Predictions | 2016 - 2020 | NCEP (National Centers for Environmental Prediction)  (accessed via University of Utah) | High real-time integration potential; requires continued data acquisition scripting & data preprocessing |
| Stage Height & Discharge at Ellicott City | 2010 - 2020 | Howard County (accessed via OneRain) | Previously used to train FLASH; gauge located on Patapsco River outside Tiber-Hudson watershed; has real-time integration potential |
| Discharge at Hollofield | 1990 - 2020 | USGS | Previously used to train FLASH; defunding risk; limited real-time integration potential |
| Discharge at Catonsville | 2010 - 2020 | USGS | Previously used to train FLASH; defunding risk; limited real-time integration potential |
| Precipitation, Convective Available Potential Energy, Surface Atmospheric Pressure | 1979 - 2020 | NLDAS-2 Primary Forcing Dataset  (accessed via Goddard Earth Sciences Data and Information Services Center [GES DISC]) | Previously used to train FLASH; spatial and temporal incongruity with STREAM model inputs |
| Soil Moisture | 1979 - 2020 | NLDAS-2 VIC Land Surface Model  (accessed via GES DISC) | Previously used to train FLASH; spatial and temporal incongruity with STREAM model inputs |