ASSESSMENT OF RAINFALL IMPACT ON ROADWAY RUNOFF USING

TRANSFORMERS

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of

the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Aayush Shukla

SPRING

2024

© 2024

Aayush Shukla

ALL RIGHTS RESERVED

ASSESSMENT OF RAINFALL IMPACT ON ROADWAY RUNOFF USING

TRANSFORMERS

A Project

by

Aayush Shukla

Approved by:

                                                                    , Committee Chair

Dr. Haiquan Chen

                                                                    , Second Reader

Dr. Anna Baynes

Date

Student: Aayush Shukla

I certify that this student has met the requirements for format contained in the University format manual, and this project is suitable for electronic submission to the library and credit is to be awarded for the project.

                                                    , Graduate Coordinator

Dr. Ying Jin  Date

Department of Computer Science

Abstractof

ASSESSMENT OF RAINFALL IMPACT ON ROADWAY RUNOFF USING

TRANSFORMERS

by  
  
Aayush Shukla

Our research aims on predicting the volume of roadway runoff volume based on rainfall. This study is significant because it allows us to estimate the quantity of pollutants, including chemicals from brake wear, vehicle emissions, and notably, polyfluoroalkyl substances (PFAS). PFAS are known for their persistence in the environment and resistance to conventional water treatment methods. By understanding and quantifying runoff, we can assess the impact of these pollutants as they are transported into larger bodies of water, posing risks to ecosystems and human health.

In this study, we employed a multivariate time series transformer model [1] to predict runoff data based on rainfall. We explored two versions of the model: one with pretraining and one without pretraining. Our research utilized a rainfall-runoff dataset provided by the California Department of Transportation. The results demonstrated that the pretrained model significantly outperformed all baseline methods in terms of RMSE (Root Mean Square Error). We've also developed a user-friendly web interface using Gradio. This interface enables users to effortlessly upload rainfall runoff data files and generate graphs comparing predicted and runoff amount.

                                              , Committee Chair

Dr. Haiquan Chen

Date

# ACKNOWLEDGEMENTS

I want to thank Dr. Haiquan Chen for his guidance on this project. He guided me in the right direction and helped me throughout my project. It was a great learning experience. I also want to thank Dr. Anna Baynes for his time as a second reader. I want to thank the Department of Computer Science for their support.

I also want to thank my parents and my family for their support. Lastly, I would like to thank my friends.

## 

TABLE OF CONTENTS

Pages

[Acknowledgements vii](#_Toc152009698)i

[List of Figures x](#_Toc152009699)

[List of Tables xi](#_Toc152009699)

Chapter

[1. INTRODUCTION 1](#_Toc152009700)

[2. DATASET 3](#_Toc152009701)

[3. Baseline models 4](#_Toc152009702)

[3.1. Transformer Model 4](#_Toc152009706)

[3.2. Long Short-Term Memory (LSTM) Model 5](#_Toc152009707)

[3.3. Bidirectional LSTM Model 5](#_Toc152009706)

[3.4. Gated Recurrent Unit (GRU) Model 6](#_Toc152009707)

[4. multivariate time series transformer 7](#_Toc152009702)  
[4.1. Multivariate Time Series Transformer Model 7](#_Toc152009706)

[5. EXPERIMENTAL VALIDATION 11](#_Toc152009705)

[5.1. Comparison between the Predicted and Actual Runoff Values 11](#_Toc152009706)

[5.2. Performance Comparison 16](#_Toc152009707)

[6. HYPERPARAMETER ANALYSIS 18](#_Toc152009708)

[7. WEB user interface 21](#_Toc152009708)

[8. CONCLUSION AND FUTURE WORK 22](#_Toc152009710)

[References 23](#_Toc152009724)

# List of Figures

Figures Page

[1. Transformer Model 4](#_Toc151988837)

[2. Multivariate Time Series Transformer Model (without Pretraining) 9](#_Toc151988838)

[3. Multivariate Time Series Transformer Model (with Pretraining) 10](#_Toc151988838)

[4. LSTM Predicted vs Actual Runoff 12](#_Toc151988839)

[5. GRU Predicted vs Actual Runoff 13](#_Toc151988840)

[6. Bidirectional LSTM Predicted vs Actual Runoff 14](#_Toc151988840)

[7. Transformer Predicted vs Actual Runoff 15](#_Toc151988840)

[8. MVTS Predicted vs Actual Runoff 16](#_Toc151988840)

[9. Weight and Biases Sweeping Result 18](#_Toc151988840)

[10. User Interface Build Using Gradio 21](#_Toc151988840)

## LIST OF TABLES

Tables Page

1. Data Statistics 3
2. MVTS Performance Comparison (24 Hour Window) 17
3. Hyperparameter Search Space 19
4. Best Hyperparameter Values for MVTS 20

## **INTRODUCTION**

Contaminants derived from sources such as vehicle exhaust and brake wear, accumulate over time and pose significant threats to ecological systems and human health. A notable concern is the way rainfall can wash these pollutants from roadways into larger water bodies, integrating toxic substances like polyfluoroalkyl substances (PFAS) into our natural and urban water cycles due to their durable and non-degradable nature.

In response to these challenges, in this study, we explore several deep learning models for the task of predicting runoff based on rainfall data, including the Multivariate Time Series Transformer model [1], Naive Transformer model [2], Long Short-Term Memory (LSTM) [3], Gated Recurrent Unit (GRU) [4], and Bidirectional LSTM models. We have used rainfall and runoff data collected from Sacramento County by California Department of Transportation.

We developed a user-friendly interface using Gradio, which enhances the accessibility and usability of our predictive model. Users will have the ability to upload their rainfall runoff data files directly through the interface. Additionally, they can easily compare and visualize the differences between the actual and predicted runoff values with an integrated chart. For further analysis, there is an option to download the predicted data in CSV format.

**Contributions of the Study:**

1. We employed the state-of-the-art Multivariate Time Series Transformer (MVTS) model [1] for time series forecasting, specifically to predict runoff data based on rainfall input.
2. Our comparative experiments demonstrated that the pretrained version of the MVTS model [1] significantly outperformed other models—including standard Transformers [2], LSTM [3], Bidirectional LSTM, and GRU [4]—in terms of RMSE.
3. We developed a web-based user interface that facilitates the prediction of runoff from rainfall data. This interface allows users to upload rainfall and runoff data and displays the predicted and actual runoff through generated graphs. Additionally, users can download the predicted data in CSV format via Gradio. Here is the link to access the Gradio UI: <https://055b4d5037185a1df7.gradio.live>

## **DATASET**

The dataset for this project is sourced from the California Department of Transportation (Caltrans). It includes detailed measurements such as rainfall amount, runoff volume, and the date and time of each event. The dataset contains data starting from 4th Jan 2010 6:30 PM till 17th Jan 2010 10:30 PM, with rainfall recorded minutely, and runoff recorded at intervals of 5-15 minutes.

In this research, we focus on developing a rainfall-runoff model for predicting runoff based on rainfall data. Upon loading the dataset from the time series file, we parse the dates in the 'Date Time' column to facilitate time-series analysis. Subsequently, the rainfall depth and runoff volume are separated into input features (X) and target labels (y) respectively. The dataset is split into training and testing sets for model training and evaluation. We have used 80% of data for training and 20% of data for testing as shown in table 1. Each training record we have used in this research has 24 hours of data window size. We have used standard scaler to normalize the range of data.

|  |  |
| --- | --- |
| **Item** | **Details** |
| Total Records | 18,240 |
| Training Data Split | 80% (14,592 records) |
| Testing Data Split | 20% (3,648 records) |

Table 1: Data Statistics

## **Baseline models**

**3.1 Transformer model**

The Transformer model [2] leverages self-attention mechanisms to parse sequences of data. This model is distinguished by its ability to manage long-range dependencies with greater efficacy than its RNN counterparts. The Transformer model [2] in our study is configured with parameters for the embedding space dimensionality, attention heads, and the feed-forward network size, among other settings. It incorporates layers of encoders and decoders as shown in figure 1, each utilizing multi-head attention and dense feed-forward networks, tailored to model the intricate relationship between rainfall intensity and runoff volume. The Transformer’s architecture is engineered to discern complex patterns within the data, serving as a potent tool for environmental phenomenon prediction.

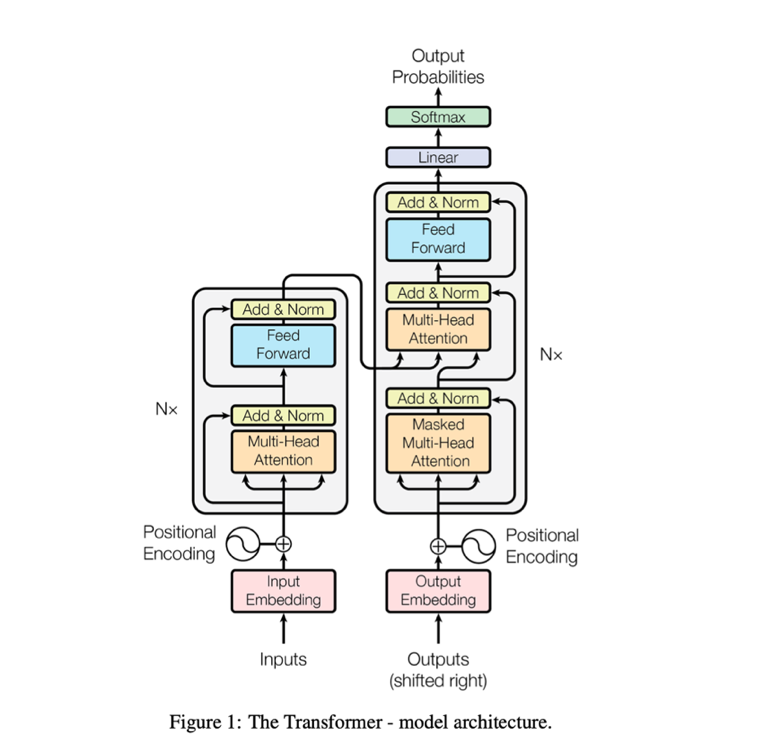


Figure 1: Transformer Model [2]

**3.2 Long Short-Term Memory (LSTM) Model**

The LSTM model [3] is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies in time series data. Our implementation comprises a sequence of three LSTM layers, each with 40 units. These layers utilize the hyperbolic tangent (tanh) activation function, allowing the model to capture both immediate and distant relationships in the data. The final layer in our LSTM model is a Dense layer with a linear activation function, tasked with producing the runoff volume prediction from the processed rainfall data. The choice of LSTM reflects its proven capability in handling complex sequence prediction tasks, where the temporal dimension plays a critical role.

**3.3 Bidirectional LSTM Model**

Based on the LSTM framework [3], the Bidirectional LSTM processes data in both forward and backward directions, providing a more holistic view of the temporal sequence. This approach enables the model to capture patterns and dependencies that may be overlooked in a unidirectional analysis. Our Bidirectional LSTM model includes a Bidirectional wrapper around an LSTM layer with 40 units, using relu activation, culminating in a Dense output layer. This model is particularly adept at forecasting tasks requiring insights into both past and future contexts within the data sequence.

**3.4 Gated Recurrent Unit (GRU) Model**

The GRU model [4] offers a simplified alternative to the LSTM, combining the forget and input gates into a single update gate, thus reducing the model complexity. Our GRU model configuration consists of a single GRU layer with 40 units, followed by two Dense layers. The first Dense layer features a rectified linear unit (relu) activation, while the second serves as the output layer with a linear activation. The GRU model's streamlined architecture not only aids in reducing training times but also maintains competitive performance, particularly in datasets where computational efficiency is paramount.

## **MULTIVARIATE TIME SERIES TRANSFORMER**

We employed a recent transformer-based deep learning model, Multivariate Time Series Transformer Model [1], to forecast the volume of runoff from rainfall events.

**4.1 Multivariate Time Series Transformer Model (MVTS)**

MVTS is based on the transformer encoder-only architecture [1]. Using only the encoder and omitting the decoder offers two main advantages. First, it eliminates the need to feed a masked output sequence to the decoder, enhancing the model's versatility for a variety of tasks. Second, by skipping the decoder, the model requires only half the parameters of a standard transformer model. This reduction in parameters not only speeds up computation but also improves learning efficiency by reducing the risk of overfitting.

Unlike traditional transformer models [2] that employ deterministic positional encodings, this model integrates fully learnable positional encodings. This modification allows the model to adapt more fluidly to the specific patterns and temporal relationships inherent in different data sequences, potentially enhancing its predictive accuracy and efficiency.

There are 2 variants of MVTS model that we have explored in this project. First is the MVTS model with Pretraining and second is MVTS model [1] without Pretraining.

**4.1.1 MVTS (Without Pretraining):** The goal of this model is to handle different tasks like regression, classification, imputation, and forecasting using just an encoder. To manage the sequential nature of time series data, this model uses positional encoding. The observations in [1] show that these positional encodings don't really interfere with the numerical data of the time series. This is likely because the encodings are set up to be learned in a way that places them in a space that doesn't overlap much with where the time series data is. This separation is easier to maintain in spaces with many dimensions, which helps keep the data's original qualities while understanding the time-related trends.

In natural language processing (NLP), transformers usually apply layer normalization following self-attention and the feed-forward steps in each encoder block. This method has shown better results than using batch normalization. However, in the approach discussed in [1] batch normalization is being used because it helps handle unusual values in time series data, a problem that doesn't occur with NLP word embeddings. We can see the architecture of this model in figure 2.

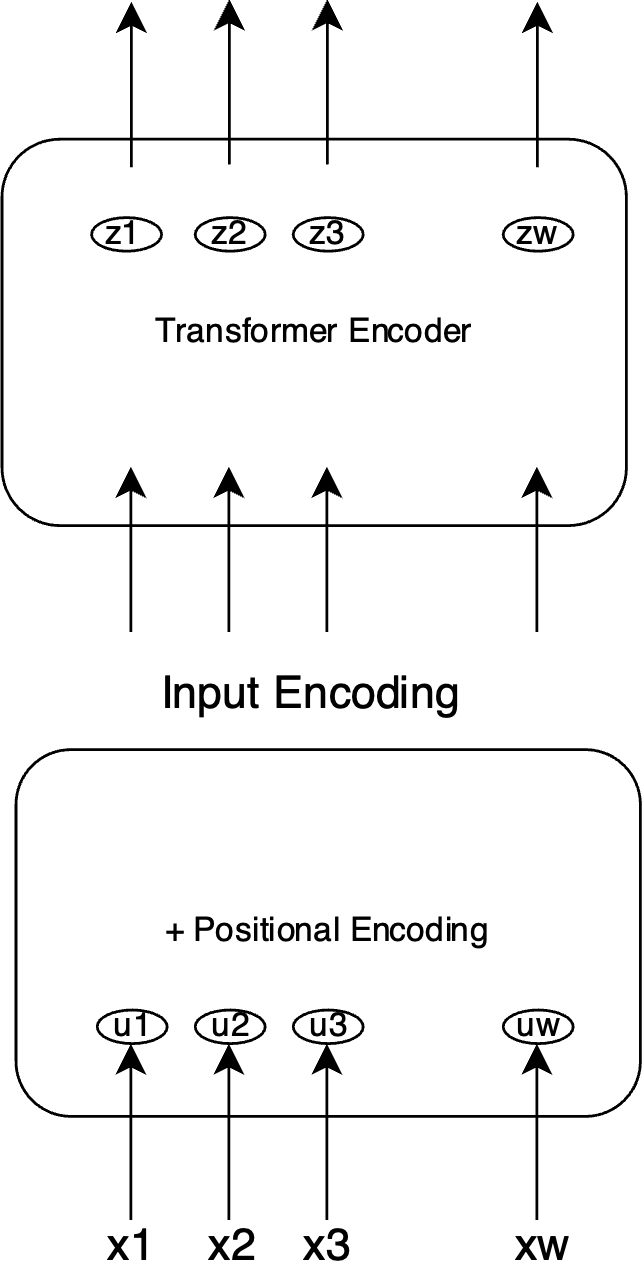


Figure 2: Multivariate Time Series Transformer Model (without Pretraining) [1]

**4.1.2 MVTS (With Pretraining):** In the pretrained MVTS model a unique binary noise mask is created for each sample in every training epoch. This mask is used by multiplying it with each data element. A portion of each column of the mask is set to zero by alternating between sequences of zeros and ones. They adjust the transition probabilities to ensure that the length of each masked segment follows a geometric distribution with a specific mean, followed by an unmasked segment. This approach is used because it allows for the approximation of small sequences in the input.

This method of input masking is distinct from the "cloze type" masking found in NLP models like BERT. In those models, a special token replaces the original word embedding, affecting the entire feature vector at those time steps. This pattern of masking is used to encourage the model to focus on both the preceding and succeeding segments of individual variables and to consider the current values of other variables in the time series. This helps the model learn the interdependencies between the variables. Architecture of this model in show in figure 3.

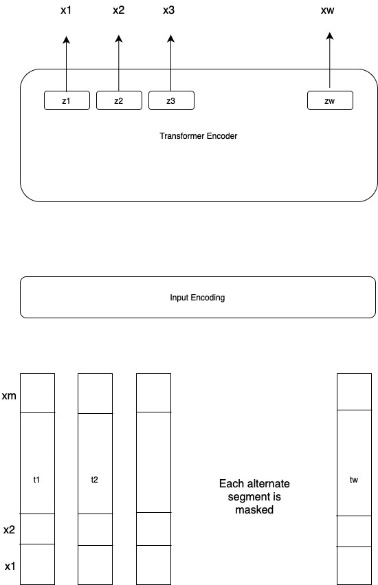


Figure 3: Multivariate Time Series Transformer Model (with Pretraining) [1]

## **experimental validation**

In this section we compared the Multivariate Time Series Transformer model with the Naive Transformer model and other baseline models. We chose the Root Mean Square Error (RMSE) as the evaluation metric.

**5.1 Comparison between the predicted and actual runoff values**

**5.1.1 LSTM:** The LSTM model we have used has 3 layers followed by a dense output later. First two LSTM layers have 40 units and uses tanh activation function. We have set return sequences as true so that output for each timestep is passed to the next layer. The third layer also have 40 units but does not return sequences which means it only gives output of last element of its processed sequence. This output serves as condensed representation of the input sequence, which is needed for final prediction. After this we have the dense later which single unit and a linear activation function, which is designed to give a continuous value as output. The model uses Adam optimizer which is efficient for handling large datasets. Figure 4 shows a comparison between actual and predicted runoff using LSTM model.

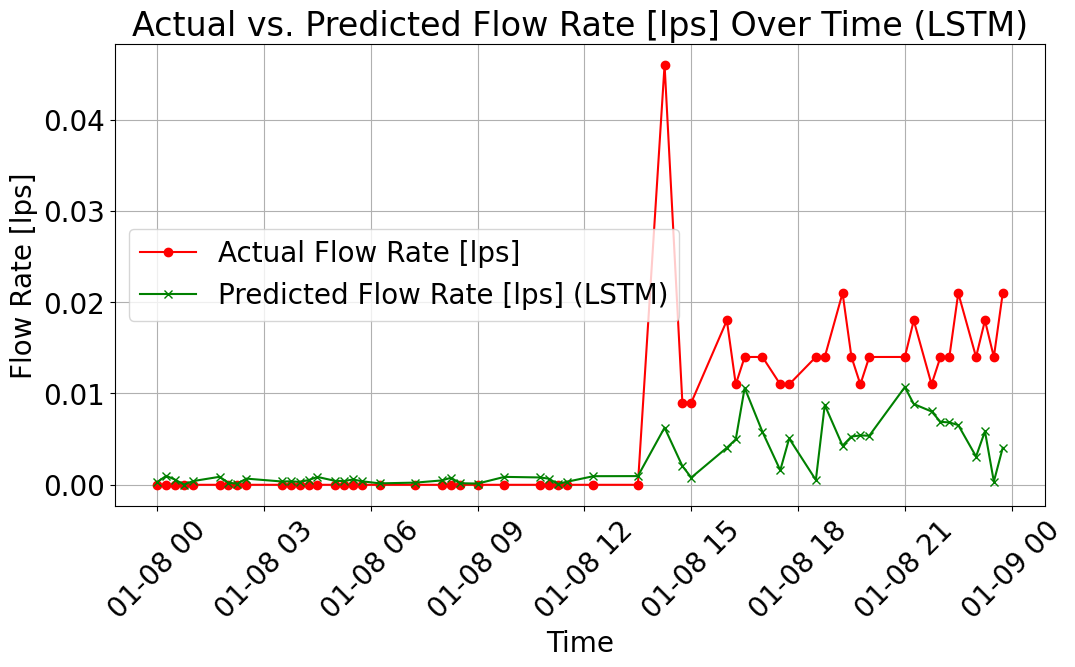


Figure 4: LSTM Predicted vs Actual Runoff

**5.1.2 GRU: T**he Gated Recurrent Unit (GRU) [4] model we have used is using rectified linear unit (relu) activation function. We have used two dense layers for this mode. The first dense later comprises 32 units and uses the relu activation function. This adds a later of non-linearity to the model which allows it to learn more complex pattern in the data. The last dense later is a single unit later which has a linear activation function. Figure 5 shows a comparison between actual and predicted runoff using GRU model.

**5.1.3 Bidirectional LSTM:** This model extends the LSTM approach by processing the data in both forward and backward directions. The Bidirectional LSTM can capture dependencies in both directions and often provides a deeper understanding of the context than unidirectional models. The model uses 40 units and a linear activation in the output layer, making it well-suited for tasks where past and future context is crucial. Figure 6 shows a comparison between actual and predicted runoff using bi-directional LSTM model.

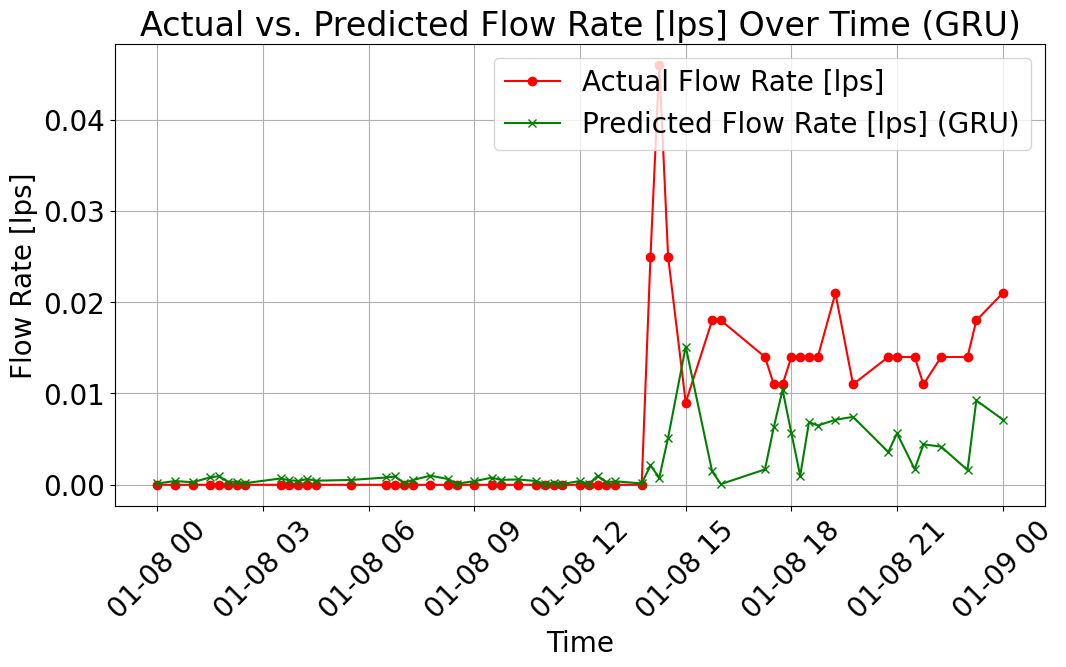


Figure 5: GRU Predicted vs Actual Runoff

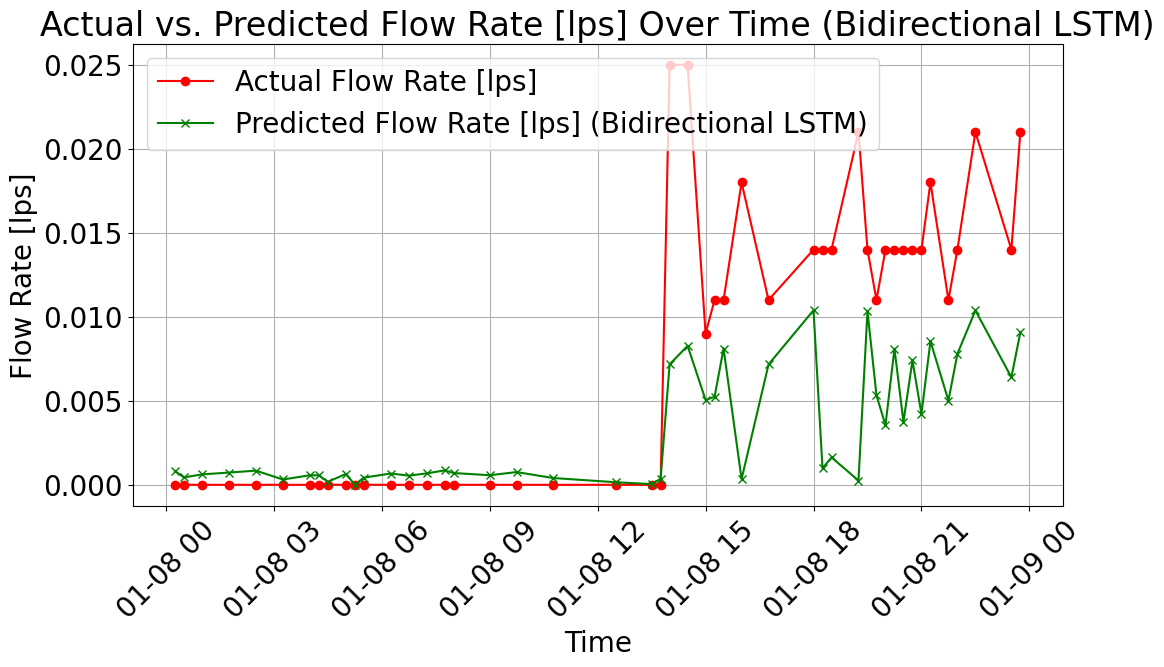


Figure 6: Bidirectional LSTM Predicted vs Actual Runoff

**5.1.4 Transformer Model:** In this research, we used and evaluated a Transformer model [2] to predict runoff data based on rainfall measurements, using TensorFlow and Keras libraries to construct a robust framework for handling time-series data. The model's architecture comprises two encoder and four decoder layers, with each encoder and decoder incorporating a 64-dimensional embedding space, dropout for regularization, layer normalization for stability, and dense layers for complexity in transformations. Key hyperparameters include four attention heads and a 512-dimensional feed-forward network. The model processes scaled training and testing data, optimizing mean squared error over 50 epochs and a batch size of 16. We tested the model focusing on the Root Mean Squared Error (RMSE) as the primary evaluation metric. Figure 7 shows a comparison between actual and predicted runoff using Transformer model.

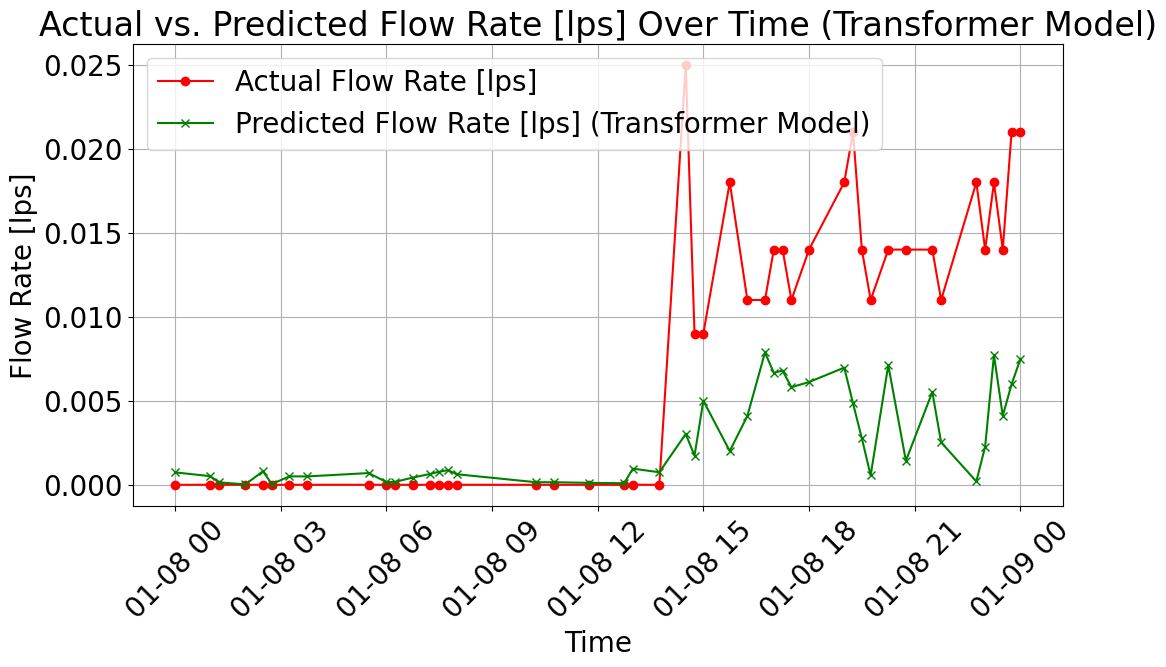


Figure 7: Transformer Predicted vs Actual Runoff

**5.1.4 MVTS Model with pretraining:** In our work with the Multivariate Time Series (MVTS) model [1], we utilized a pre-trained model as a foundational base, extending its training for an additional 700 epochs to tailor it more closely to our specific dataset and task. We fine-tuned the learning rate to 0.001 and opted for the RAdam optimizer, known for its robust performance in various settings. The model was configured with a dimensionality (d\_model) of 64, and uniquely, we employed learnable position encodings to enhance the model's ability to capture temporal relationships. The batch size was set to 32, and the GELU activation function was chosen for its effectiveness in deep learning models. These adjustments culminated in achieving an impressive Root Mean Square Error (RMSE) of 0.0131, underscoring the efficacy of our model tuning and training strategy. Figure 8 shows a comparison between actual and predicted runoff using MVTS model.

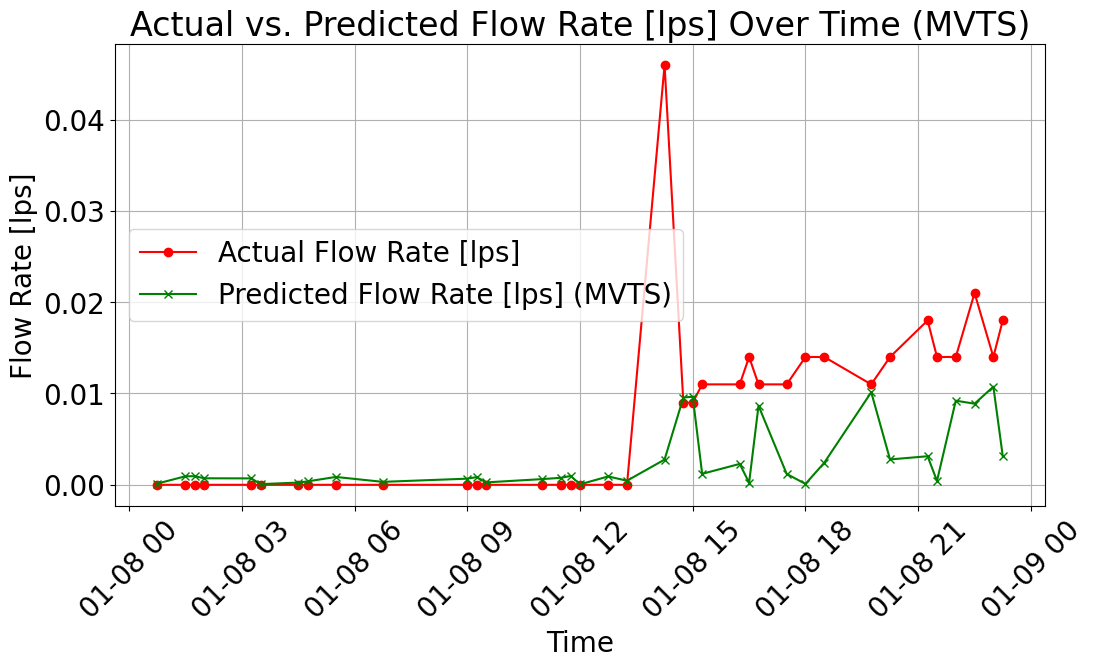


Figure 8: MVTS Predicted vs Actual Runoff

**5.2 Performance Comparison**

As demonstrated in Table 2 of our analysis, the Multivariate Time Series (MVTS) model, after extensive tuning and additional training, achieved the best Root Mean Square Error (RMSE) score of 0.0131. This indicates a highly successful optimization process tailored to our specific dataset and analytical goals. Comparatively, the pre-trained MVTS model, without the additional epochs of training and fine-tuning, performed as the second-best configuration, yielding an RMSE of 0.017. This comparison highlights the significant improvement gained through our optimization efforts, illustrating the value of model customization and further training in enhancing predictive accuracy.

|  |  |
| --- | --- |
| **Model** | **Best RMSE** |
| LSTM | 0.0294 |
| GRU | 0.0294 |
| Bi-directional LSTM | 0.0294 |
| Naïve Transformer | 0.0232 |
| MVTS without Pretraining | 0.0174 |
| MVTS with Pretraining | 0.0132 |

Table 2: Models Performance Comparison (24-hour window)

## **HYPERPARAMETER** **Tuning**

In the context of optimizing a Multivariate Time Series Transformer model using Weights & Biases (W&B), the search space for each hyperparameter is defined through a Bayesian optimization method, as specified in the sweep configuration. This approach efficiently searches the set of hyperparameter that results in minimum loss. The parameters set in the configuration include learning rate, number of encoders and decoders, model dimensions (d\_model), number of attention heads (num\_heads), feedforward network dimension (ff\_dim), batch size, and the number of training epochs. Each parameter is provided with a discrete set of values to be tested. For instance, learning rates of 0.01, 0.001, and 0.0001; number of encoders and decoders each set at 1, 2, and 4 respectively; and similarly detailed options for other parameters. The 10 best performing hyperparameter tuning results are show in Figure 9.

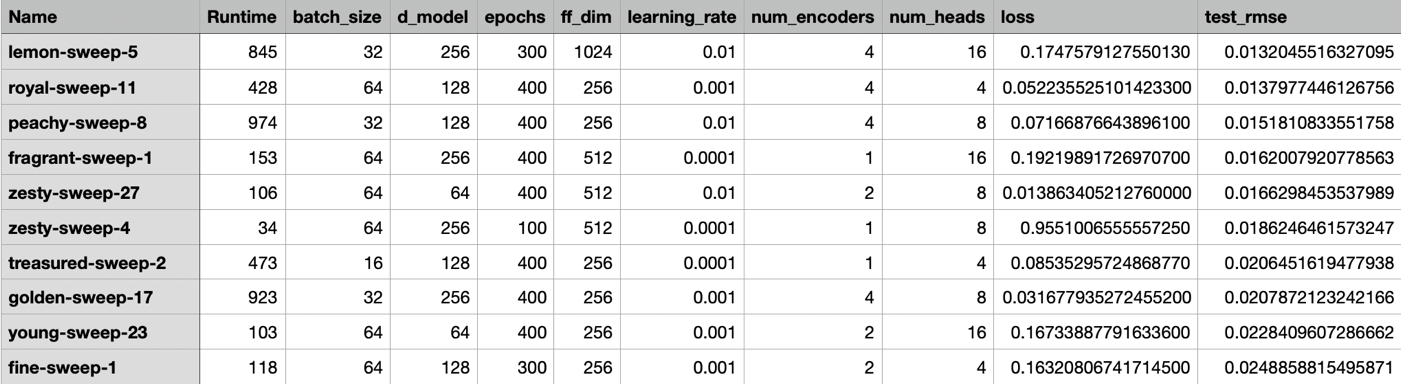


Figure 9: Weight and Biases Sweeping Result

|  |  |
| --- | --- |
| **Hyperparameter** | **Search Space** |
| Batch size | 16, 32, 64 |
| Model Dimension | 64, 128, 256 |
| Feed forward network dimension | 256, 512, 1024 |
| Learning Rate | 0.01, 0.001, 0.0001 |
| Number of encoders | 1, 2, 4 |
| Number of attention heads | 4, 8, 16 |
| Epochs | 200, 300, 400, 500 |

Table 3: Hyperparameter Search Space

In Table 3, we present the comprehensive search space utilized for the hyperparameter tuning of our model, conducted using the Weights and Biases sweeping tool. The search space includes a variety of settings to optimize model performance. Batch sizes were tested at three levels: 16, 32, and 64. The model dimension was varied among 64, 128, and 256, allowing us to assess the impact of increasing complexity on model accuracy. The dimension of the feed-forward network was also examined at three scales: 256, 512, and 1024. Learning rates were considered at 0.01, 0.001, and 0.0001 to fine-tune the speed and stability of the convergence. The architecture was further tweaked by adjusting the number of encoders (1, 2, or 4) and the number of attention heads (4, 8, or 16), to optimize processing and attention mechanisms. Lastly, the model was evaluated over several training durations, specifically at 200, 300, 400, and 500 epochs. This extensive tuning process aims to identify the optimal combination of parameters that enhances model performance across different scenarios.

The configuration that yielded the best results in the optimization process with Weights & Biases for the Multivariate Time Series Transformer model is shown in table 4 It utilized a batch size of 32 and a model dimension (d\_model) of 256. The model was trained over 300 epochs, with a feedforward network dimension (ff\_dim) of 1024. An initial learning rate of 0.01 was set, and the architecture included 4 encoders, with a substantial 16 attention heads. The model recorded RMSE (Root Mean Square Error) value of approximately 0.0132. This configuration and its results highlight the efficiency of the Bayesian optimization method in navigating the hyperparameter space to find an effective set of values that contribute to the model’s predictive performance.

|  |  |
| --- | --- |
| **Hyperparameter** | **Value** |
| Batch size | 32 |
| Model Dimension | 256 |
| Feed forward network dimension | 1024 |
| Learning Rate | 0.01 |
| Number of encoders | 4 |
| Number of attention heads | 16 |
| RMSE | 0.0132 |

Table 4: Best Hyperparameter Values for MVTS

1. **WEB USER INTERFACE USING GRADIO**

As shown in figure 10 we provide an intuitive web interface where the user can easily upload their rainfall runoff data and generate a comparative graph of predicted versus actual runoff values. The user will have the flexibility to choose from various predictive models, including LSTM [3], GRU [4], Bidirectional LSTM, and Transformer Model [2], based on specific data analysis. The platform also allows to define the analysis period by selecting a start and end date. Outputs are conveniently available as downloadable CSV files containing actual and predicted runoff, alongside a Matplotlib graph for direct comparison. Additionally, we offer downloadable templates and sample datasets to facilitate the testing and optimization of our models.

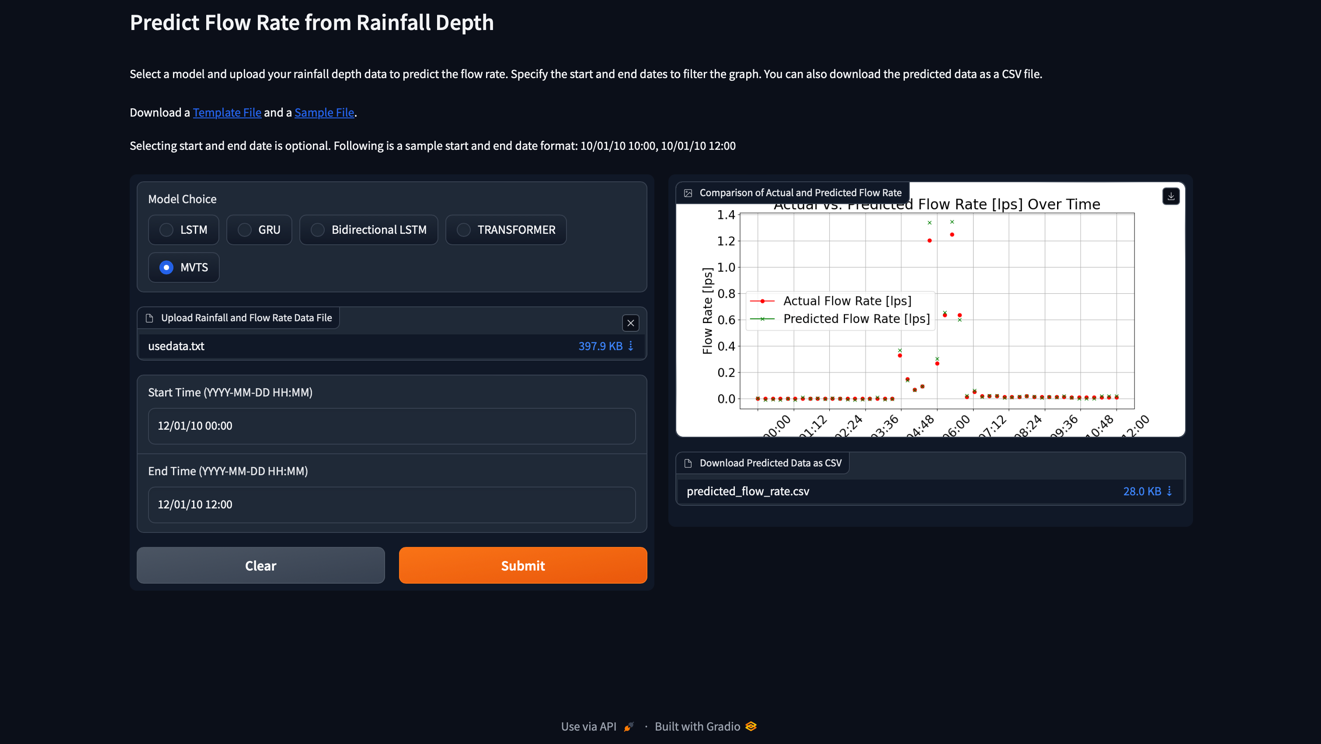


Figure 10: User Interface Build Using Gradio

1. **CONCLUSION AND FUTURE WORK**

This research represents a significant step forward in the prediction of urban rainfall-runoff, a critical component in the management of urban water systems and flood mitigation strategies. By leveraging advanced machine learning models, including LSTM [3], GRU [4], Bidirectional LSTM, Naïve Transformer [2] and the Multivariate Time Series Transformer model [1], we have demonstrated the potential of these technologies to accurately forecast runoff volumes in real-time. Our comparative analysis, grounded in the use of RMSE as a benchmark for performance evaluation, highlights the strengths and limitations of each model, providing valuable insights into their practical application in environmental science.

## **REFERENCES**

1. Zerveas, George, et al. "A Transformer-based Framework for Multivariate Time Series Representation Learning." arXiv preprint arXiv:2010.02803 (2020). Available at: <https://arxiv.org/abs/2010.02803>.
2. Vaswani, Ashish, et al. "Attention is All You Need." arXiv preprint arXiv:1706.03762 (2017). Available at: <https://arxiv.org/abs/1706.03762>.
3. Staudemeyer, R. C., & Morris, E. R. (2019, September 12). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks. arXiv.org. <https://arxiv.org/abs/1909.09586>
4. Dey, R., Salem, F. M., Circuits, Systems, and Neural Networks (CSANN) LAB, & Department of Electrical and Computer Engineering, Michigan State University. (n.d.). Gate-Variants of Gated Recurrent Unit (GRU) neural networks. In Department of Electrical and Computer Engineering, Michigan State University [Journal-article]. <https://arxiv.org/pdf/1701.05923.pdf>
5. Carleton, C. & Office of Water Programs at California State University, Sacramento. (2015). QUANTIFYING REPRESENTATIVE SAMPLING USING a HYDROLOGIC <https://owp.csus.edu/research/papers/papers/quantifying-representative-sampling-using-a-hydrologic-analysis-tool-carleton-2015.pdf>
6. Ouma, Y. O., Cheruyot, R., & Wachera, A. N. (2021). Rainfall and runoff time-series trend analysis using LSTM recurrent neural network and wavelet neural network with satellite-based meteorological data: case study of Nzoia hydrologic basin. *Complex & Intelligent Systems*, *8*(1), 213–236. <https://doi.org/10.1007/s40747-021-00365-2>
7. Chadalawada, J., Herath, H. M. V. V., & Babovic, V. (2020). Hydrologically Informed Machine Learning for Rainfall‐Runoff Modeling: A Genetic Programming‐Based toolkit for automatic model induction. *Water Resources Research*, *56*(4). <https://doi.org/10.1029/2019wr026933>