CNNs VERSUS LSTMs FOR TIME SERIES FORECASTING

by

Bidur Prasad Bhurtel

B.E., Tribhuvan University, 2017

A Thesis
Submitted in Partial Fulfillment of the Requirements for the
Master of Science Degree

Department of Electrical, Computer and Biomedical Engineering
in the Graduate School
Southern Illinois University Carbondale
May 2021

THESIS APPROVAL

CNNs VERSUS LSTMs FOR TIME SERIES FORECASTING

by

Bidur Prasad Bhurtel

A Thesis Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Master of Science

in the field of Electrical and Computer Engineering

Approved by:

Dr. Lalit Gupta, Chair

Dr. Spyros Tragoudas

Dr. Mohammad Sayeh

Graduate School
Southern Illinois University Carbondale
March 30, 2021

AN ABSTRACT OF THE THESIS OF

Bidur Prasad Bhurtel, for the Master of Science degree in Electrical and Computer Engineering, presented on March 30, 2021, at Southern Illinois University Carbondale.

TITLE: CNNs VERSUS LSTMs FOR TIME SERIES FORECASTING

MAJOR PROFESSOR: Dr. Lalit Gupta

The goal of this thesis is to compare the performances of long short-term memory (LSTM) recurrent neural networks and feedforward convolution neural networks (CNNs) in time series forecasting. The forecasting problem focuses on predicting the future values of a time series using the current and a set of previous (lagged) values of the time series. LSTMs are used extensively in time series forecasting problems because they are specifically designed to process sequential and temporal data. CNNs on the other hand are not designed to process such sequential data. Although CNNs appear to be a poor choice for time series forecasting, it would be informative to compare the performances of CNNs and LSTMs by using exactly the same set of current and previous values of several combinations of multiple time-series. The specific forecasting problem considered involves predicting the outflow for a creek sub-basin using outflow, temperature, and precipitation time series consisting of 185,544 hourly datapoints. The Granger causality (GC) test is used to confirm that temperature and precipitation Granger cause outflow and should, therefore, be helpful in predicting outflow. The GC test also provides information on the lag required to influence outflow prediction. The forecasting problem is divided into developing 3 distinct types of models: one-hour forecast models, 2-hour forecast models, and a one-hour and 2-hour extreme-event forecast models. The one-hour forecast models are trained to predict the next-hour outflow using the current and a set of previous values of the time series. The 2-hour forecast models are trained to predict the outflow 2 hours ahead using the current and previous values of the time series. A variation of the two-hour model is trained with the previous, current, and the next-hour prediction of the one-hour model. The extreme-event forecast models are trained only with segments of the time series containing extreme events. The performance of the LSTM and CNN implementations of the models are compared objectively using the mean square error and subjectively by comparing the predictions visually. The results from various combinations of the creek sub-basin time series show that the CNN models outperform the LSTM models. These results are quite unexpected given that CNNs appear to be a poor choice for time series forecasting whereas LSTMs are a good choice. The primary reason for the CNN models yielding superior performance is that through the choice of appropriate filters, CNNs are capable of generating complex features which are coupled simultaneously across time series predictor variables and across time lags in the series of convolution layers. However, it cannot be concluded that CNNs will, in general, outperform LSTMs without testing the models on a large ensemble of diverse data sets.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my academic and thesis adviser, Dr. Lalit Gupta, for his invaluable guidance throughout the completion of my master's study at SIUC. I would like to thank Dr. Tragoudas and Dr. Sayeh, for graciously agreeing to serve as my thesis committee members and for their helpful comments and suggestions.

DEDICATION

To my loving mom and dad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Thesis Goal

The goal of this thesis is to compare the performances of convolution neural networks (CNNs) and long short-term memory (LSTM) neural networks for time series forecasting. The recurrent structure of LSTMs make them an obvious choice for predicting future values of time series. The feed forward CNNs on the other hand are not specifically designed to process time series data. However, CNNs have been shown to yield outstanding performance in numerous classification and regression problems. It would, therefore, be informative to develop CNN forecasting models and compare them with the LSTM forecasting models on the same data set. The specific forecasting problem in this study involves predicting the outflow for a creek sub-basin using three time series: outflow, precipitation, and temperature. The precipitation and temperature are included in forecasting because they are shown to Granger-cause outflow. That is, the inclusion of precipitation and temperature in the outflow forecasting model should improve the prediction of outflow.

Forecasting can be one-step or multi-step. One step forecasting involves predicting one observation step ahead while multi-step forecasting involves predicting several steps ahead. The forecasting problems in this study are divided into developing 3 distinct types of models: one-step forecast models, 2-step forecast models, and a one-step and 2-step extreme-event forecast models. The step size is one hour, therefore, the models are referred in terms of hours rather than step sizes. The one-hour forecast models are trained to predict the next-hour outflow using the current and a set of previous values of the time series. The two-hour forecast models are trained to predict the outflow 2 hours ahead using the current and previous values of the time series. A variation of the two-hour model is trained with the previous, current, and the next-hour prediction of the one-hour model. The extreme-event forecast models are retrained only with segments of the time series containing extreme events. The LSTM and CNN implementations of three types of model are trained and tested using various combinations of the three time series. The performances of the LSTM and CNN models are compared objectively using the mean square error and subjectively by comparing the predictions visually.

1.2. The Time Series Forecasting Problem

By definition, a time series is a sequence of observations acquired sequentially in time. Forecasting in time series involves developing models that fit historical observations and then using the models to predict future observations. Examples of time series forecasting include predicting future values of stock prices, weather, gas prices, electricity demand, birth rates, and number of individuals infected by a virus [1, 2, 3]. Various classical forecasting models have been developed which include: Autoregressive (AR) [4, 5], Moving Average (MA) [6], Autoregressive Moving Average (ARMA) [5, 7], Vector Autoregression (VAR) [8], and Vector Autoregression Moving Average with Exogenous Regressors (VARMAX) [9, 10].

1.3. LSTM Neural Networks

In recent years, attention has focused on using recurrent neural networks such as LSTMs for time series forecasting [11, 12]. For time series forecasting, remembering the trends of past events are very important to make accurate predictions. Recurrent neural networks (RNNs) address this issue with feedback loops that allow previous information to persist. A drawback with regular recurrent neural networks is that when the number of loops grows, RNNs suffers from the vanishing gradient problem [13, 14]. That is, the gradients that are at the deeper stage of the RNN layer will have less impact on the prediction or forecasting problem. The LSTM neural network was designed to overcome this problem. Furthermore, LSTMs also solve complex, artificial long-time-lag tasks that have never been solved by previous recurrent networks algorithms [15]. A weakness in the original LSTM [16] which did not have the forget gate was that the internal state could grow indefinitely which could cause the network to break down eventually [17]. The current LSTM cell has three gates. The gates in LSTM are the sigmoid activation function which outputs a value between zero to one. The significance of using a sigmoid function for gates instead of any other function is that the sigmoid function gives us only positive values, so that the network can get clear idea if it needs to either keep or discard a particular feature. The equation of sigmoid function is:

$$sig(x) = \frac{1}{1 + e^{-x}}$$

And the graph of sigmoid function is shown in Figure 1.

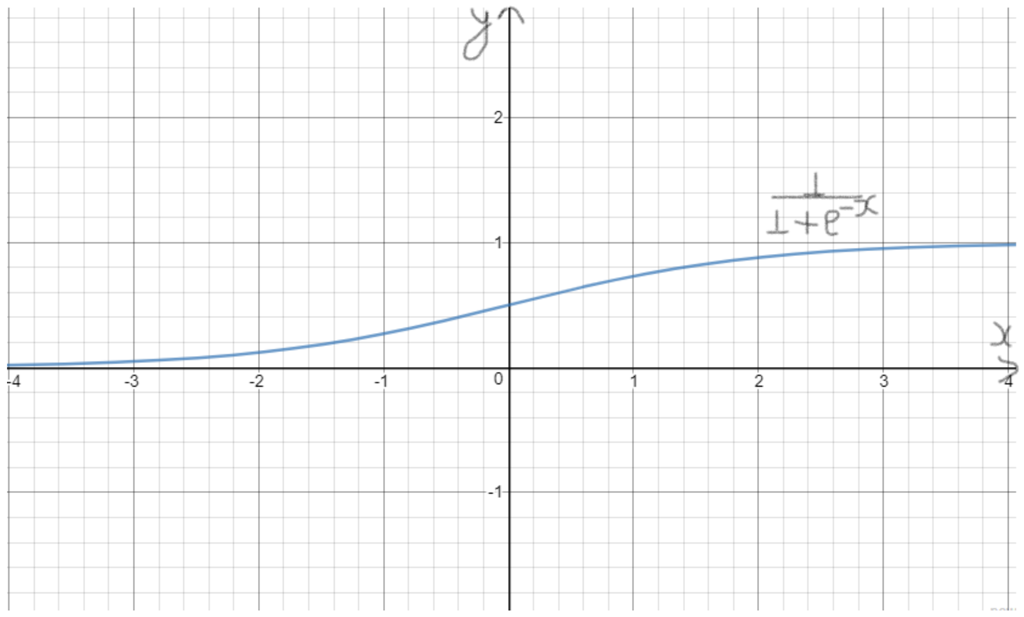


Figure 1.1: Graph of Sigmoid Function

The equations used in LSTMs for the three gates are:

$$i_t = \sigma (w_i [h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma (w_f [h_{t-1}, x_t] + b_f)$$
$$O_t = \sigma (w_o [h_{t-1}, x_t] + b_o)$$

where, i_t, f_t, and O_t represent the input gate, forget gate and output gate respectively. The biases of the respective gates are b_i, b_f and b_o. The sigmoid function is represented by σ, h_{t-1} is the output of the previous LSTM block, x_t is the current input and w_i, w_f, w_o are the weights of the respective gates. The operations of the three gates are as follows:

i. Input gate: The function of the input gate is to instruct the network which new information should be stored in the cell state. The cell state is the long-term memory of the LSTM which is modified by the forget gate and adjusted by the input gate. The equations for cell state, candidate cell state and hidden state are:

$$C'_t = \tanh (w_c [h_{t-1} , x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * C'_t$$
$$h_t = O_t * \tanh(C_t)$$

where C_t is the cell state at time t, C'_t is the candidate cell state at time t, and h_t is the hidden state.

ii. Forget gate: This gate is responsible for instructing the network about which information is to be discarded or retained from the cell state.

iii. Output gate: The output gate is used to provide the activation to the final output of the LSTM network. The cell state is passed through a hyperbolic tanh function to filter the cell state values between -1 and 1.

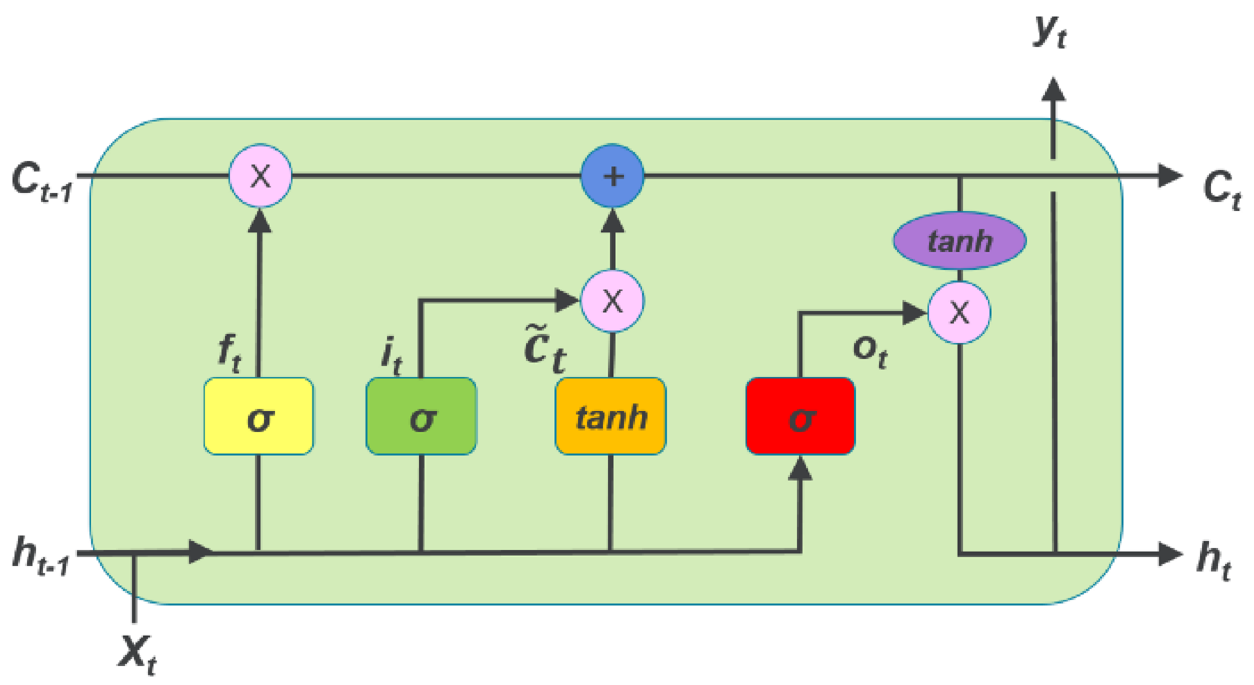


Figure source: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af

Figure 1.2: LSTM block with input, forget and output gate

The LSTM cell consisting of the three gates is illustrated in Figure 2.

1.4. <u>CNN Neural Networks</u>

The architecture of CNN is somewhat similar to the connection patterns of neurons in the human brain and individual neurons in CNN responds to activation of certain area of input matrix passed on to it. CNNs can find features through spatial and temporal dependencies in an input matrix. They can learn the relation between rows and columns in an input data which will help in understanding the high degree of complexity in input data. CNNs can be trained faster than regular feed forward layers since they have relatively less trainable parameters. The convolutional layer, which are central to CNN, performs convolution of kernel and input data that results in a feature map, which reveals some distinct feature on the input data as shown in Figure 3.

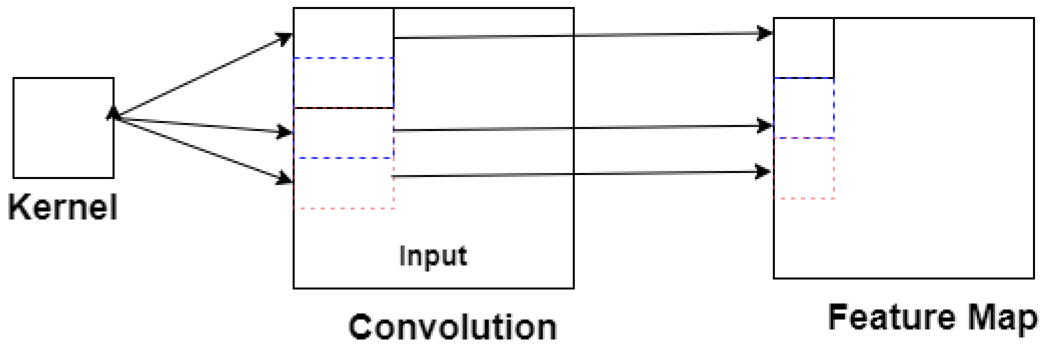
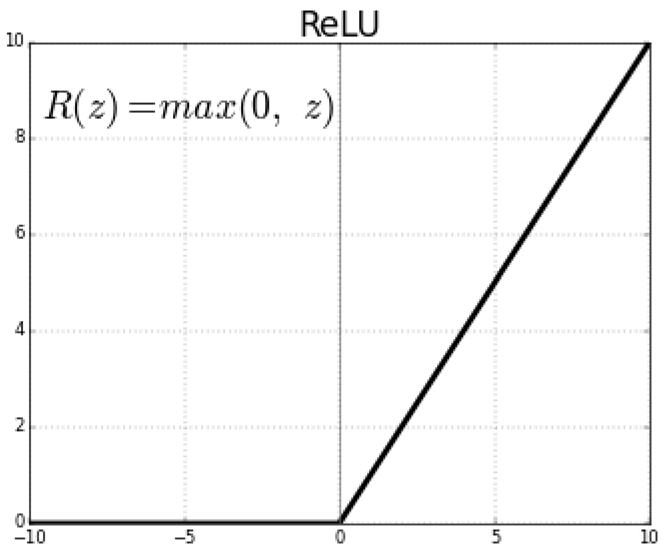


Figure 1.3: Extraction of feature map by convolution of kernel with input

The kernel is applied onto the input by repeated overlapping and multiplying the corresponding overlapped elements and summing all the multiplied results. This summed result occupies the corresponding place on the feature map. The convolution step helps to extract some unique feature based on kernel values, which are learnable parameters in the model. Convolution can result in two types of results in terms of dimensionality based on the type of padding used. A "valid" padding would result in feature matrix with reduced dimensionality in compared to the input after convolution while "same" padding would result in a feature matrix with the same dimensionality as the input.

The feature matrix produced by convolutional layer is passed to a non-linear activation function because the response of an arbitrary element in the network with only linear elements can be described simply as linear sum of the weights and inputs [18]. So, to exploit full capability of a multilayered networks, we introduce non-linearity in neural networks to predict a nonlinear output through nonlinear combinations of trainable weights in hidden layers and input. The non-linear activation that we use in our study is ReLU. Figure 4 shows the graph and equation for ReLU activation function.



F*igure 1.4: ReLU activation function*

The ReLU activation is used in all layers of our model except the final dense layer. The final dense layer uses linear activation functions. Linear activation is simply the multiplication of weight matrix with feature matrix summed with the bias. It is equivalent to applying no activation function at all.

Pooling layers are used to down sample the results from convolutional layers. These layers will create a new set of pooled feature matrices by operating upon each feature matrix separately and they always reduce the size of feature matrix. Two common types of pooling are average pooling and max pooling. In average pooling, we slide across the feature map and compute the average value of each patch on the feature matrix while in max pooling maximum value from a patch is taken for new set of pooled feature matrices. Pooling helps to make a model invariant to small translations of the input, so if the input changes by a small amount, there is no change in the values of most of the pooled outputs [19]. Figure 5 is an example of max pooling layer used in CNN network.

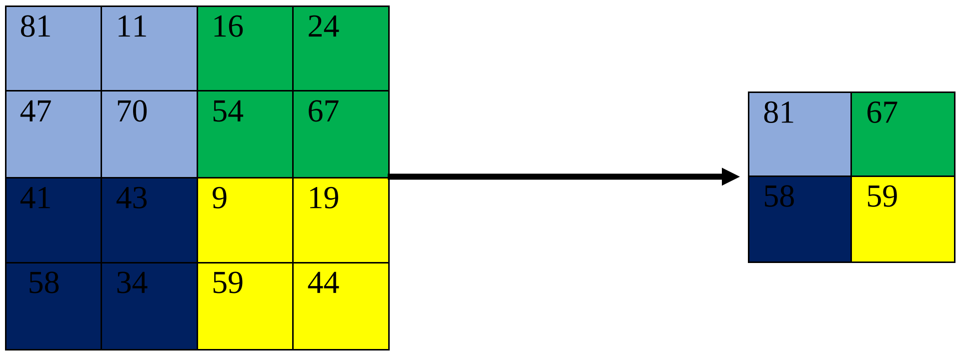


Figure 1.5: Max pooling with 2x2 pooling filter

The output from the final pooling or convolutional layer is flattened and fed to the fully connected layers in CNN models. Each neuron in fully connected layer will calculate

$$f(b + w * x)$$

In the above equation, x is the input vector, w is the weight matrix, b is the bias vector and f is the activation function. The final layer of feed forward layer acts as an output layer and it has a linear activation function for forecasting problem. Figure 6 is an illustration of a typical CNN consisting of convolution, pooling, and a fully connected layer.

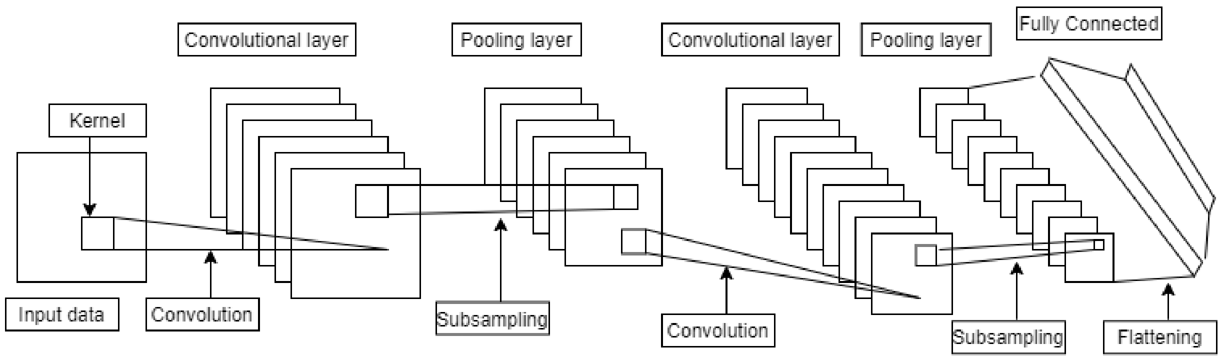


Figure 1.6: CNN model example

CHAPTER 2

METHODOLOGY

2.1. Data

The time series used to train and test the forecasting models were hourly precipitation, temperature, and outflow for the Dry Creek Sub-basin within the Tuolumne River Basin, located in Sonoma and Mendecino Counties in California. Hourly precipitation and temperature were collected over the 1980-2001 time period across the Tuolumne Basin from the gridded meteorological dataset [20]. Corresponding hourly outflow was collected from the USGS National Water Information System (https://waterdata.usgs.gov/ca/nwis/uv?site_no=11465350). The complete data set contained 8 time-series of Dates, Year, Month, Hour, Hours, Precipitation, Temperature and Outflow with each time-series containing 185,544 datapoints. A snapshot of the data is shown in Table 1. The first 5 columns were discarded because the data in those columns do not carry any useful information for forecasting outflow. Table 2 describes the statistics of the three time-series selected.

Table 2.1: Snapshot of the Dry Creek Sub-basin time series

Dates	Year	Month	Day	Hours	Precipitation	Temperature	Outflow
12/3/1980	1980	12	3	16:00:00	0.010295	51.89	85
12/3/1980	1980	12	3	17:00:00	0.020591	52.952	87
12/3/1980	1980	12	3	18:00:00	0.012869	54.014	88
12/3/1980	1980	12	3	19:00:00	0.012869	54.644	89
12/3/1980	1980	12	3	20:00:00	0.038607	55.292	90
12/3/1980	1980	12	3	21:00:00	0.03346	55.922	93
12/3/1980	1980	12	3	22:00:00	0.020591	55.832	97
12/3/1980	1980	12	3	23:00:00	0.021877	55.742	101
12/4/1980	1980	12	4	0:00:00	0.01496	55.652	104
12/4/1980	1980	12	4	1:00:00	0.023686	54.896	107
12/4/1980	1980	12	4	2:00:00	0.034906	54.122	109
12/4/1980	1980	12	4	3:00:00	0.029296	53.366	111

Table 2.2: Statistics of the 3 time series used to forecast outflow

	Precipitation	Temperature	Outflow
count	185544	185544	185544
mean	0.001575	64.515827	15.481724
std	0.012426	15.337548	60.61091
min	0	17.978	0
25%	0	52.862	0
50%	0	62.204	1
75%	0	74.264	5
max	1.184073	116.06	1920

2.2. Data Preparation

2.2.1. Data Normalization

As reported in [21], convergence is usually faster if the average of each input variable over the training set is close to zero. Therefore, each time series was scaled to have values in the interval [0,1] using the library function MinMaxScalar from scikit-learn. The transformation is given by the equation:

$$X\_std = \frac{(X - X.min)}{(X.max - X.min)}$$

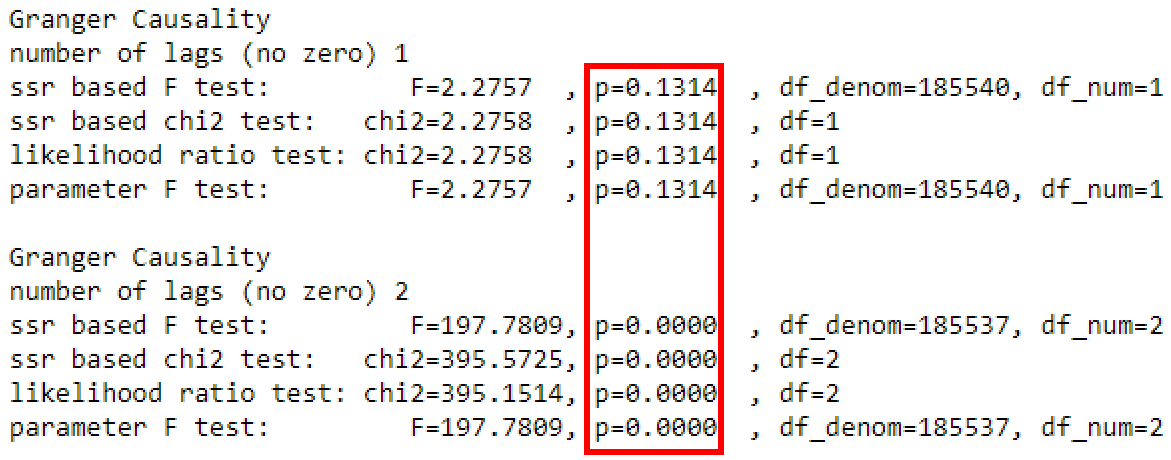
$$X\_scaled = X\_std * (max - min) + min$$

where min, max = feature range(0,1).

2.2.2 Granger Causality Test

Granger Causality Test (GCT) helps in determining whether one time series is useful in forecasting another time series [22]. The investigation if the two time-series have some form of causal relationship could give us an idea of the time-series which would be useful in forecasting.

Although we have simple mathematical tool to check the correlation between two variables of multivariate autoregressive models, correlation does not necessarily conclude that one variable is the cause for other. GCT not only gives us if the relationship between two variables is causal or not, but it also helps us to determine the correct number of time-lag that we should use so that the effect of one variable can be seen onto another. In Granger Causality Test, if the p-value for a certain lag is less than a predefined value, which is 0.05 in our case, we conclude that one time-series is Granger causing another. For our data, we found that the time-series of Outflow is caused by the time-series of both Precipitation and Temperature after of lag of 2. Figure 7 below shows that after a lag of 2, outflow follows the time-series Temperature. Similar test was done for Precipitation and found that outflow is also caused by Precipitation. So, we could use any appropriate lag that is greater than 2. For our study, we choose 18.



```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=2.2757  , p=0.1314  , df_denom=185540, df_num=1
ssr based chi2 test:   chi2=2.2758  , p=0.1314  , df=1
likelihood ratio test: chi2=2.2758  , p=0.1314  , df=1
parameter F test:         F=2.2757  , p=0.1314  , df_denom=185540, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=197.7809, p=0.0000  , df_denom=185537, df_num=2
ssr based chi2 test:   chi2=395.5725, p=0.0000  , df=2
likelihood ratio test: chi2=395.1514, p=0.0000  , df=2
parameter F test:         F=197.7809, p=0.0000  , df_denom=185537, df_num=2
```

Figure 2.1: Result of Granger Causality Test

2.2.3 <u>One-Hour Forecast Models</u>

These models are used to predict the outflow for the next hour using current and previous values of the three time series in various combinations. The following models were developed under this category:

a. predict next-hour outflow using current and 17 previous outflow values.

b. predict next-hour outflow using current and 17 previous outflow and precipitation values.

c. predict next-hour outflow using current and 17 previous outflow and temperature values.

d. predict next-hour outflow using current and 17 previous outflow, precipitation, and temperature values.

e. predict next-hour outflow using current and 17 previous precipitation and temperature values.

2.2.4. Two-Hour Forecast Models

These models are used to predict the outflow 2-hours later using current and previous values of the three time series in various combinations. The following models were developed under this category:

a. predict 2-hour later outflow using current and 17 previous outflow values.

b. predict 2-hour later outflow using current and 17 previous outflow and precipitation values.

c. predict 2-hour later outflow using current and 17 previous outflow and temperature values.

d. predict 2-hour later outflow using current and 17 previous outflow, precipitation, and temperature values.

e. predict 2-hour later outflow using current and 17 previous precipitation and temperature value.

Another set of 2-hour later prediction models were developed by augmenting the above models with the next-hour forecast of the outflow. The lag, therefore, increased from 18 to 19 in these models. Yet another set models we developed by augmenting the models with the next-hour predicted values of outflow, precipitation, and temperature. We have demonstrated below how the input vector for univariate and multivariate time series was formed for 1-hour and 2-hour predictions.

Univariate Time Series Forecasting (example)

Time Series X: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, \ldots, x_n, \ldots, x_N$

Assume Lag = 4 (4 past observations to predict the next observation)

Input	1-Step Output	2-step Output	CNN Filter	CNN Filter Output
x_1			(4x1)	(1x1)
x_2				
x_3				
x_4	$\hat{x}_5$	$\hat{x}_6$		
x_2			(4x1)	(1x1)
x_3				
x_4				
x_5	$\hat{x}_6$	$\hat{x}_7$		
x_3			(4x1)	(1x1)
x_4				
x_5				
x_6	$\hat{x}_7$	$\hat{x}_8$		
.	.	.		
.	.	.		
.	.	.		
.	.	.		

Multivariate Time Series Forecasting (example)

Time Series X: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, \ldots, x_n, \ldots, x_N$

Time Series Y: $y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, \ldots, y_n, \ldots, y_N$

Time Series Z: $z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, \ldots, z_n, \ldots, z_N$

Assume Lag = 4 (4 past observations to predict the next observation)

Input	[1-Step Output]	[2-step Output]	[Filter1]	[Filter1 Output]	[Filter 2]	[Filter 2 Output]
$[x_1, y_1, z_1]$			(1x3)	(1x1)	(4x1)	(1x1)
$[x_2, y_2, z_2]$						
$[x_3, y_3, z_3]$						
$[x_4, y_4, z_4]$	$\hat{x}_5$	$\hat{x}_6$				
$[x_2, y_2, z_2]$			(1x3)	(1x1)	(4x1)	(1x1)
$[x_3, y_3, z_3]$						
$[x_4, y_4, z_4]$						
$[x_5, y_5, z_5]$	$\hat{x}_6$	$\hat{x}_7$				
$[x_3, y_3, z_3]$			(1x3)	(1x1)	(4x1)	(1x1)
$[x_4, y_4, z_4]$						
$[x_5, y_5, z_5]$						
$[x_6, y_6, z_6]$	$\hat{x}_7$	$\hat{x}_8$				
.	.	.				
.	.	.				
.	.	.				
.	.	.				
.	.	.				
.	.	.				
.	.	.				
.	.	.				
.	.	.				

2-Step Prediction augmented with 1-Step Prediction (example)

Assume Lag = 4 (4 past observations to predict the next observation)

Input	2-Step Output	Filter1	Filter1 Output	Filter 2	Filter2 Output
$[x_1, y_1, z_1]$		(1x3)	(1x1)	(5x1)	(1x1)
$[x_2, y_2, z_2]$					
$[x_3, y_3, z_3]$					

$[x_4, y_4, z_4]$					
$[\hat{x}_5, \hat{y}_5, \hat{z}_5]$	$\hat{x}_6$				
$[x_2, y_2, z_2]$		(1x3)	(1x1)	(5x1)	(1x1)
$[x_3, y_3, z_3]$					
$[x_4, y_4, z_4]$					
$[x_5, y_5, z_5]$					
$[\hat{x}_5, \hat{y}_5, \hat{z}_5]$	$\hat{x}_7$				
$[x_3, y_3, z_3]$		(1x3)	(1x1)	(5x1)	(1x1)
$[x_4, y_4, z_4]$					
$[x_5, y_5, z_5]$					
$[x_6, y_6, z_6]$					
$[\hat{x}_7, \hat{y}_7, \hat{z}_7]$	$\hat{x}_8$				
.		.	.		
.		.	.		
.		.	.		
.		.	.		

2.2.5 Extreme-Event Forecast Models

These models are used to predict extreme outflow for the next hour using current and previous values of the three time series in various combinations. An extreme-event occurs if the outflow takes a value greater than a visually determined threshold equal to 250. The following models were developed under this category:

a. predict 1-hour and 2-hour extreme outflow using current and 17 previous outflow values.

b. predict 1-hour and 2-hour extreme outflow using current and 17 previous outflow and precipitation values.

c. predict 1-hour and 2-hour extreme outflow using current and 17 previous outflow and temperature values.

d. predict 1-hour and 2-hour extreme outflow using current and 17 previous outflow, precipitation, and temperature values.

e. predict 1-hour and 2-hour extreme outflow using current and 17 previous precipitation and temperature values.

2.3. Deep Learning Models Setup

All variations of the three types of forecasting models were implemented using LSTMs and CNNs. The hyperparameters for both the LSTM and CNN models were determined from a validation set. In order to keep the comparisons fare across the LSTM models, the same architecture was used in the LSTM models. Similarly, the same architecture was used across the CNN models.

2.3.1. LSTM

The LSTM model had three LSTM layers and three dropout layers after each LSTM layer. The dropout layers were used for regularization. The dropout layer randomly sets input to zero with a frequency of given rate at each step during training time. The last LSTM layer flattens the data since we do not return sequence in the last LSTM layer. After the last dropout layer, we use three dense layers, one of which is used as output layer of the model. Figure 8 shows an example of a three-feature input architecture of the LSTM model. All LSTM models that we used in the study had a similar architecture with the only difference being the input data shape.

Layer 1: The first LSTM layer had 16 hidden units and the input shape to this layer was (datasize)x(lag size)x(number of input features). Since the return sequences is set to be true, the output from this layer was (datasize)x(lag size)x(16). If the return is set to false, the output shape would be (datasize)x(16)x(1). This is because the return sequences return the hidden state output for each input time step. The dropout layer after this layer had a rate of 0.2. The dropout layer does not change the input shape.

Layer 2: The second LSTM layer had 32 hidden units and the input shape to this layer was (datasize)x(lag size)x(16) and since the return sequences is true for this layer, the output from this layer has dimension (datasize)x(lag size)x(32). The dropout layer after this layer also had a rate of 0.2.

Layer 3: The final LSTM layer had 64 hidden units and the input shape to this layer was (datasize)x(lag size)x(32). Unlike the previous two LSTM layers, the final LSTM layer had return sequences set to false and the output from this layer was (datasize)x(64)x(1). The dropout layer after this layer also had a rate of 0.2.

Dense1: The first dense layer had 128 hidden units with ReLU activation functions. The input shape to this layer was (datasize)x(64).

Dense2: The second dense layer had 64 hidden units with ReLU activation functions. The input shape to this layer was (datasize)x(128).

Dense3: The final dense layer acts as the output layer of the network. The number of hidden units in the final dense layer is set according to the requirement of the output shape. The activation functions was linear for this layer and the input shape to this layer was (datasize)x(64).

2.3.2. <u>CNN</u>

Just as the LSTM architectures, the CNN models consisted of three layers. The three layers were followed by a flattening layer and a three-layer dense network. Figure 9 shows an example of a three-feature input architecture of the CNN model. All CNN models that were used in the study had similar architectures with the only difference being the input data shape. Pooling layers were not included in the architectures to ensure there was no loss of information. No dropout was used in the CNN models because dropout is generally ineffective in CNNs because the networks have a relatively small number of parameters. Experiments were conducted to confirm that there was no improvement in the performance by including dropouts in the CNN models developed in this study.

Layer 1: Layer 1 used one-dimensional convolutional layer with 16 filters of size 1x3 in order to extract features across different time-series. Same padding and ReLU activation functions were used in all the layers. The input data shape to this layer varied according to the number of features that were used and output shape was (datasize)x(lag size)x16.

Layer 2: The second convolution layer used 32 (3x1) one-dimensional filters. The input size to this layer was (datasize)x(lag size)x16 and output size was (datasize)x(lag size)x32.

Layer 3: This layer used 64 (3x1) filters and input shape to this layer was (datasize)x(lag size)x32 and output shape was (datasize)x(lag size)x64.

Flattening Layer: The flattening layer reshapes the output of last convolution layer into a vector.

Dense1: The first dense layer had 1152 hidden units with ReLU activation functions. 1152 hidden units were selected because this was the dimension of the flattened output.

Dense2: The second dense layer had half of the number of hidden units of the first dense layer and also used ReLU activation functions.

Dense3: The final dense layer acted as the output layer of our CNN model. The number of hidden units in this layer differed according to the required output shape.

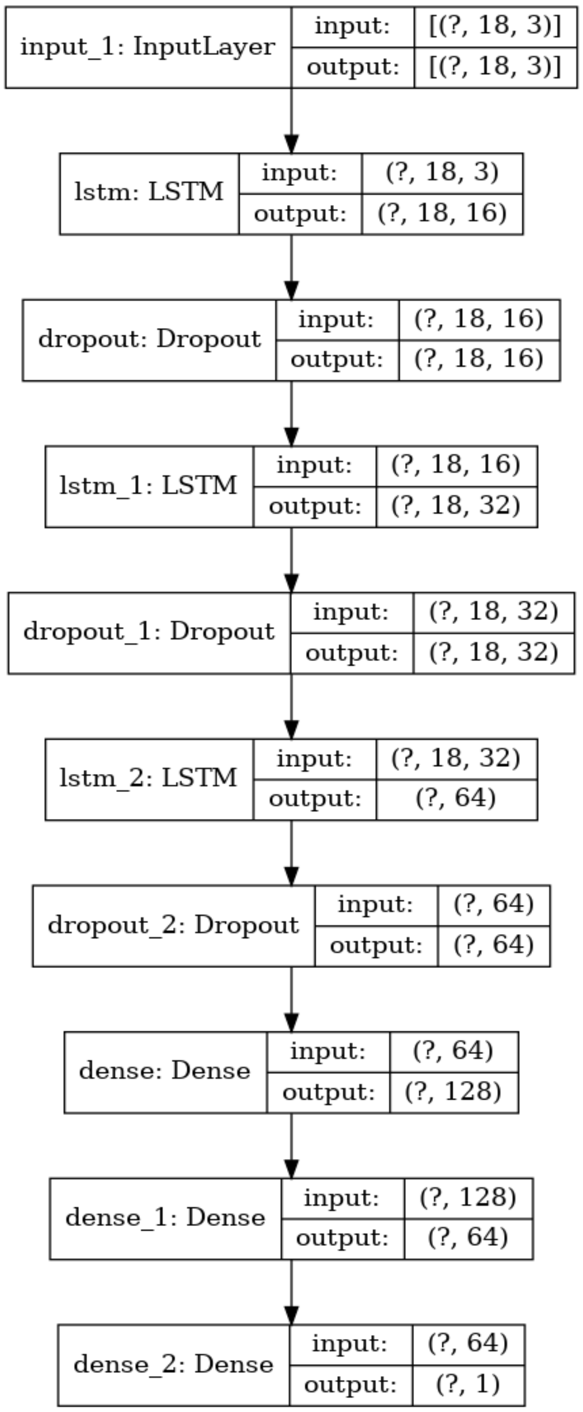


Figure 2.2: LSTM model for three feature input

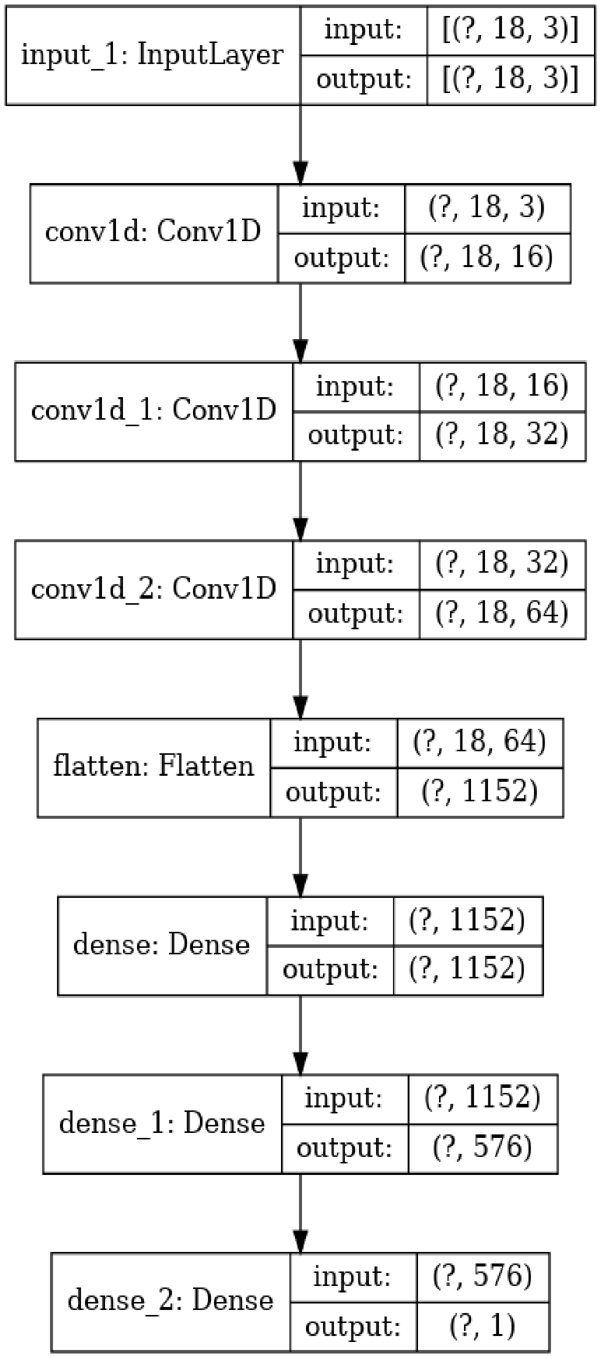


Figure 2.3: CNN model for three feature input

2.4. Libraries Used

Programming Language: Python

Environment: Jupyter Notebook

Python Libraries: Numpy, Matplotlib, Pandas, Time, Tqdm

Machine Learning Libraries: Tensorflow, Keras, Scikit-learn

Hardware: Intel(R) Core i3-8130U CPU @ 2.20GHz

In addition to our own hardware, we used google colaboratory through which we had GPU access for faster training time.

CHAPTER 3

RESULTS

The data that was used to train the models was divided into 3 sets: a training set consisting of the first 70% of the data points, a validation set consisting of the next 10% of the data points, and a test set consisting of the remaining 20% of the data points. The hyperparameters for the models were determined from the validation set. The prediction accuracy of each model was measured objectively using the mean square error between the predicted and actual value. The prediction was also judged subjectively by plotting the predicted values against the true values. The tables in this section list the average mean square errors over 20 iterations for each model.

3.1. <u>1-Hour Prediction Results</u>

The 1-hour prediction results for outflow using different predictor inputs are summarized in Table 3.

Table 3.1: Mean square errors of the 1-hour prediction models

	PREDICTOR INPUTS				
Model/	**Outflow**	**Precipitation, Outflow**	**Temperature, Outflow**	**Precipitation, Temperature**	**Precipitation, Temperature, Outflow**
CNN	31.05	30.45	41.2	4229.61	**29.3**
LSTM	70.33	93.88	88.57	4061.08	81.42

It is interesting to observe that except for one case (precipitation and temperature inputs without outflow input), the CNN models outperformed the LSTM models. The best result is highlighted in red font.

3.2. Hour-2 Prediction Results

The 2-hour results are summarized in Table 4. The models under “None” do not use any 1-hour predictions. The models under the “All Predicted” correspond to filling in the 1-hour predictions of each input predictor to forecast the 2-hour outflow. It is important to observe that, except for one case which does not use outflow as an input, the CNN models outperform the LSTM models. In general, adding the 1-hour predictions do not help improve the 2-hour predictions, except when the predicted outflow is used in the CNN model. When we use only outflow as input, CNN model using 1-hour predictions improve the 2-hour predictions, but the LSTM model does not.

Table 3.2: Mean square errors for the 2-hour prediction models

Predictor Inputs	**None Predicted**		**All Predicted**	
	CNN	**LSTM**	**CNN**	**LSTM**
Flow	54.71	145.4	47.48	161.9
Prec, Flow	49.41	125.3	70.56	156.25
Temp, Flow	53.55	122.21	77.51	182.04
Prec, Temp	2741.51	4121.75	1652.27	1609.44
Prec, Temp, Flow	56.13	126.4	86.82	140.7

3.3. Extreme Value Predictions

The 1-hour and 2-hour extreme event predictions are summarized in Table 5. The relatively high mean square errors indicate that both the CNN and LSTM models are poor at predicting extreme events.

Table 3.3: Mean square errors value of 1-hour and 2-hour predictions of extreme events

Model/ Features	Hour 1		Hour 2	
	CNN	**LSTM**	**CNN**	**LSTM**
Flow	299532.89	299507.04	299538.23	299481.97
Pre, Flow	299529.13	299539.18	299636.95	299556.13
Tem, Flow	299537.04	299519.78	299543.28	299501.95
Pre, Tem	299690.79	299667.95	299526.38	29949.94
Pre, Tem, Flow	299534.63	299541.07	299582.12	299552.89

3.4. Plots of Predicted vs Original outflow Values

Figure 10 to Figure 23 shows plots of predicted vs original outflow value for different training variations that we ran for our study.

3.4.1. <u>1-Hour Predictions</u>

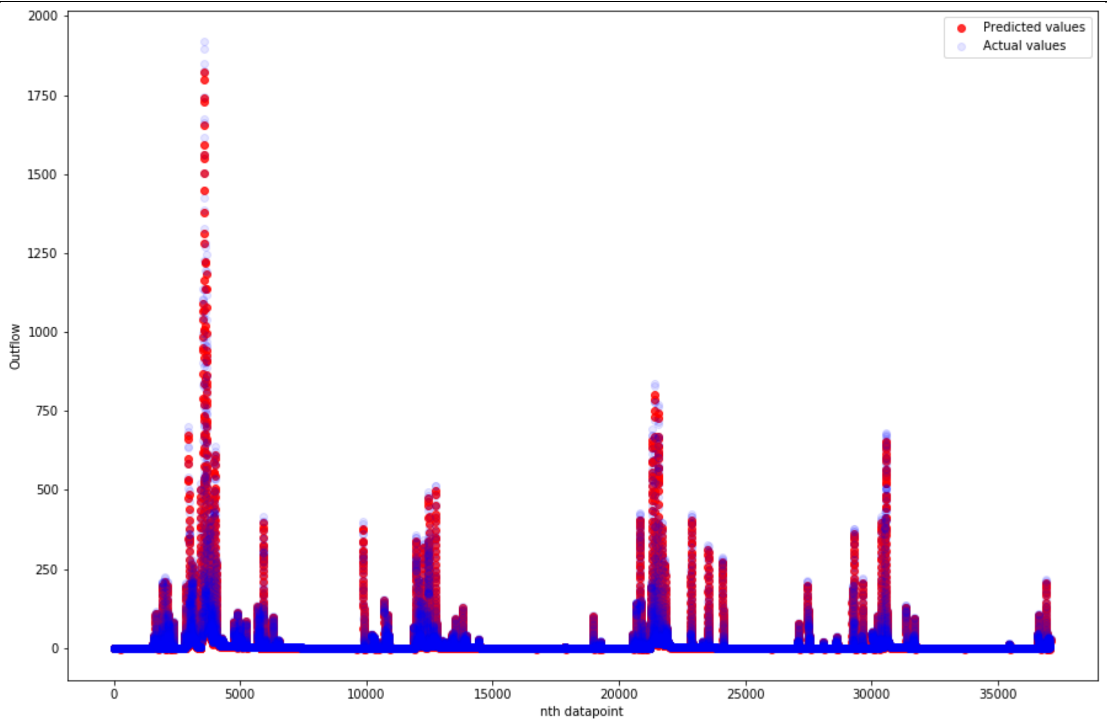
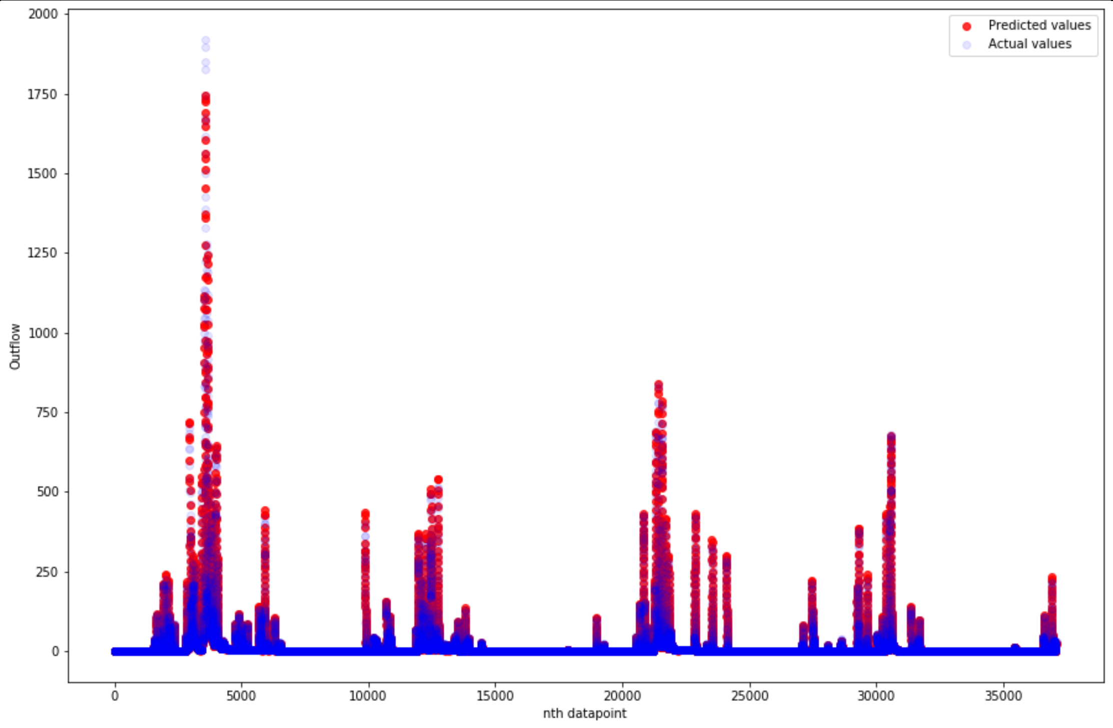
CNN	
LSTM	

Figure 3.1: 1-hour comparisons using Outflow for the input

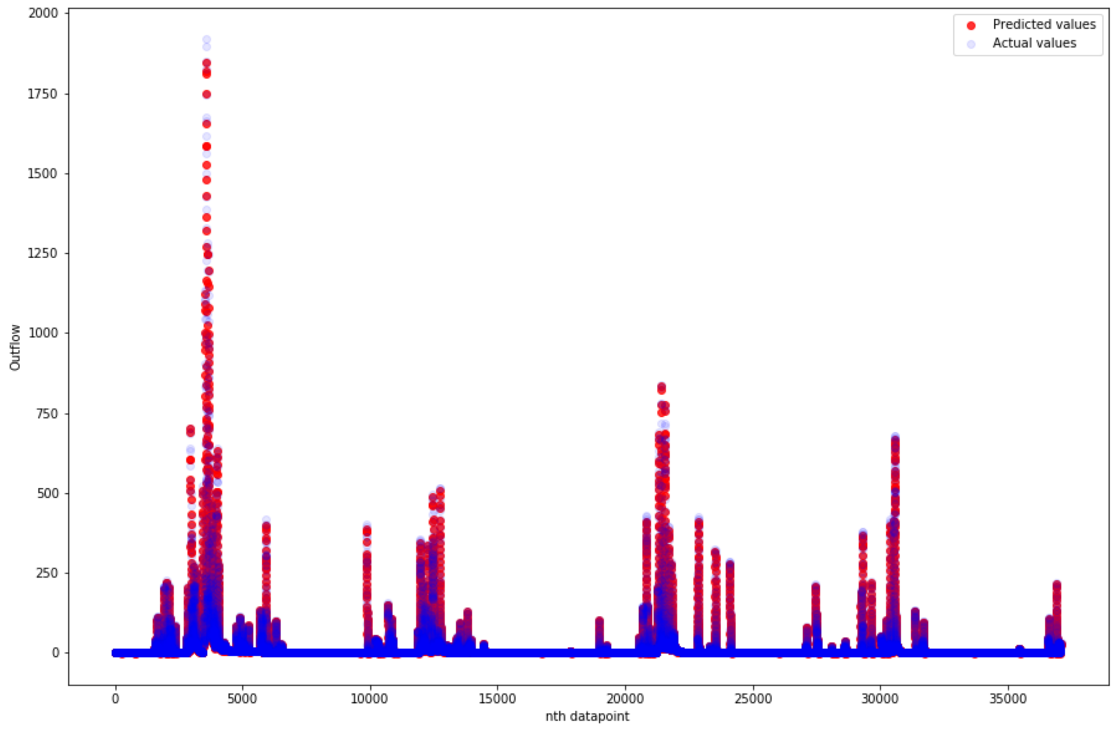
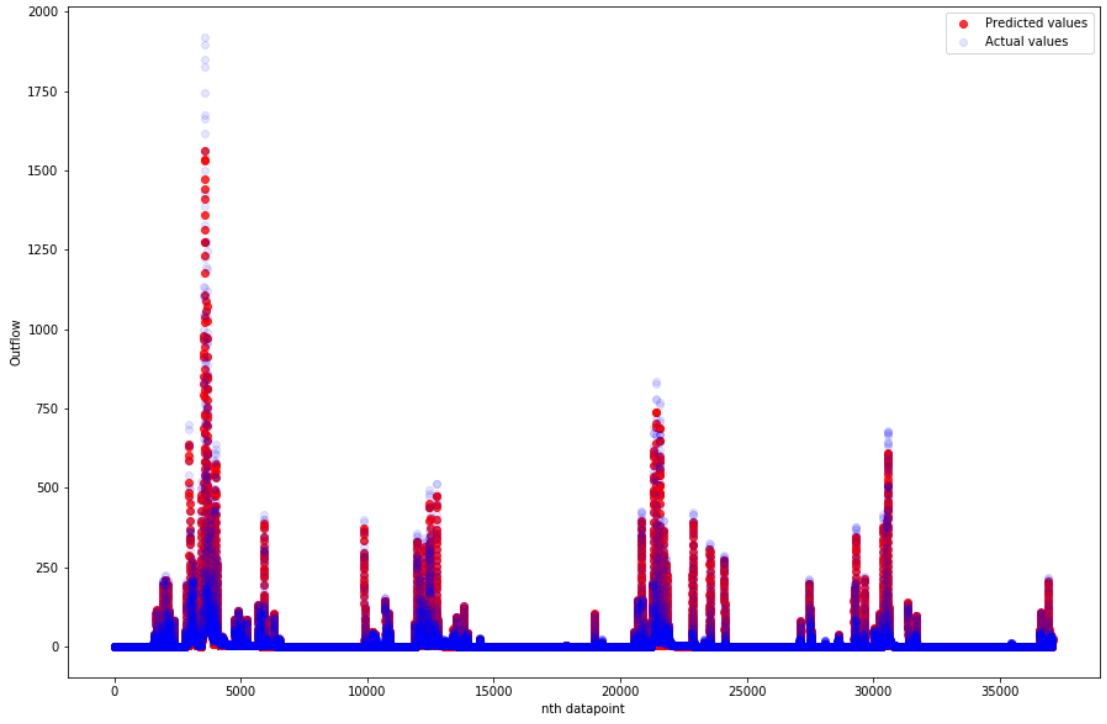
CNN	
LSTM	

Figure 3.2: 1-hour comparisons Precipitation and Outflow for inputs

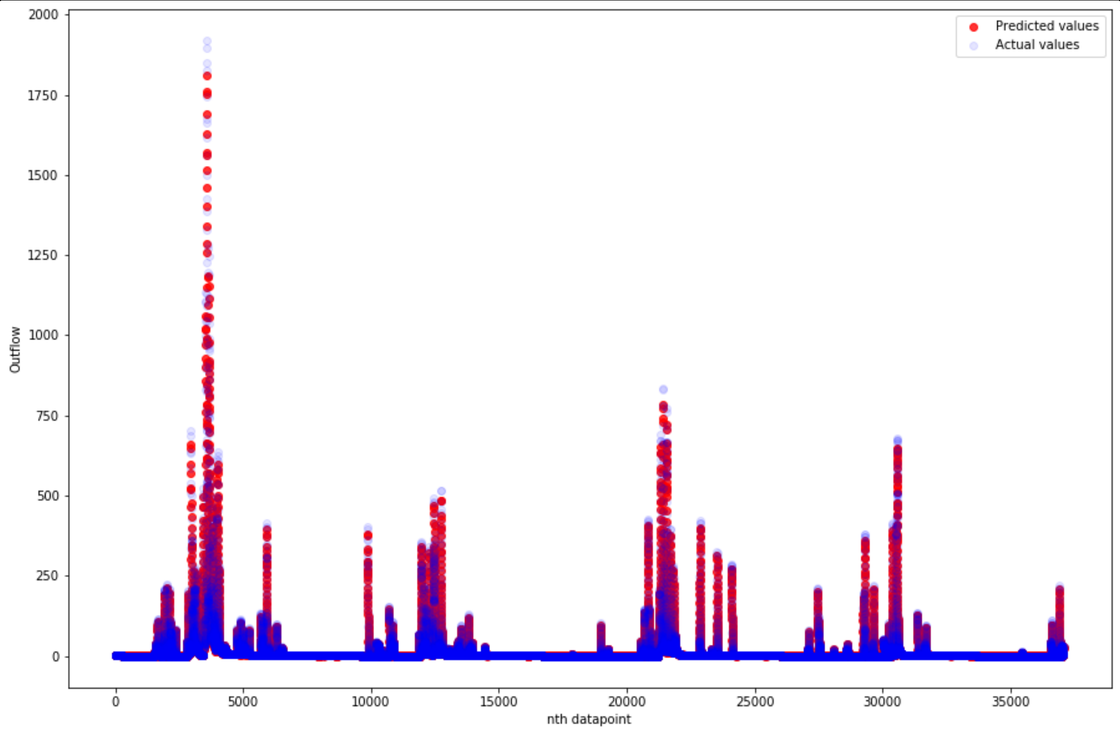
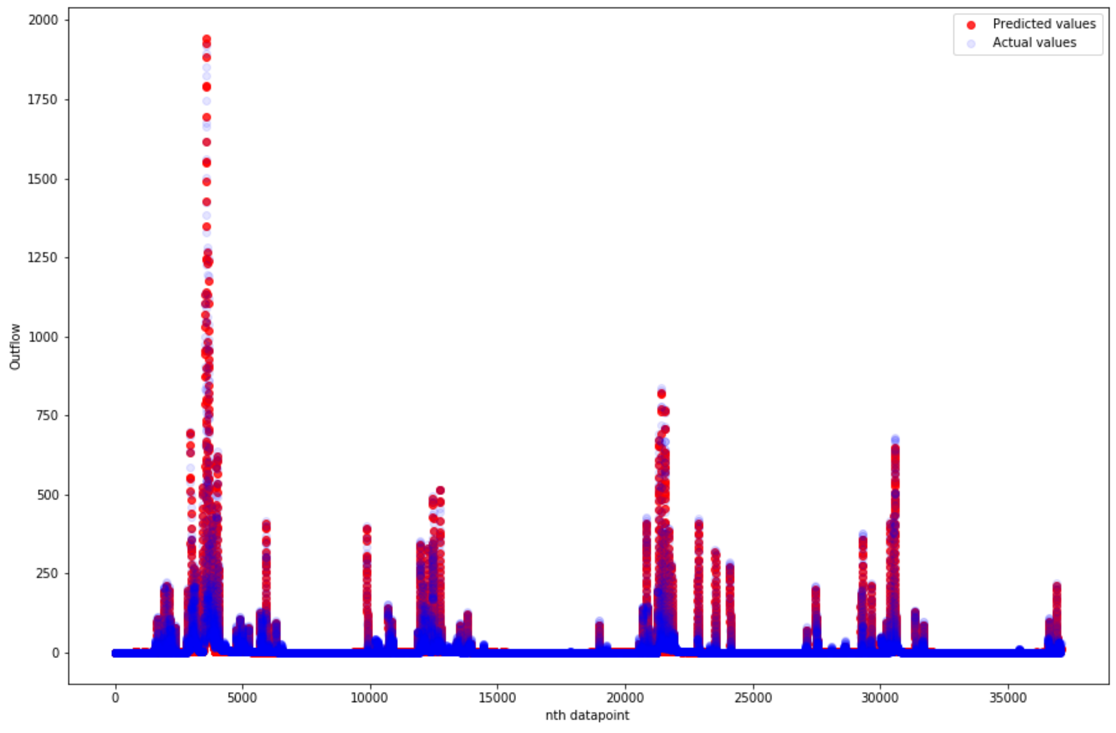
CNN	
LSTM	

Figure 3.3: 1-hour comparisons using Temperature and Outflow for inputs

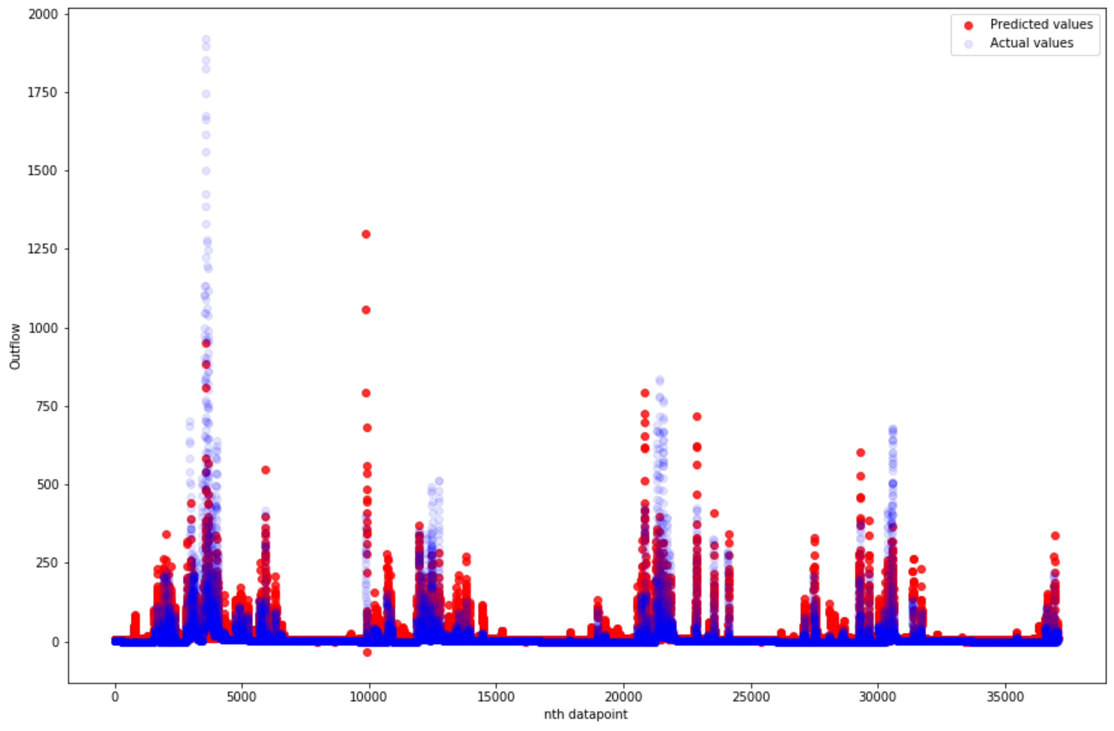
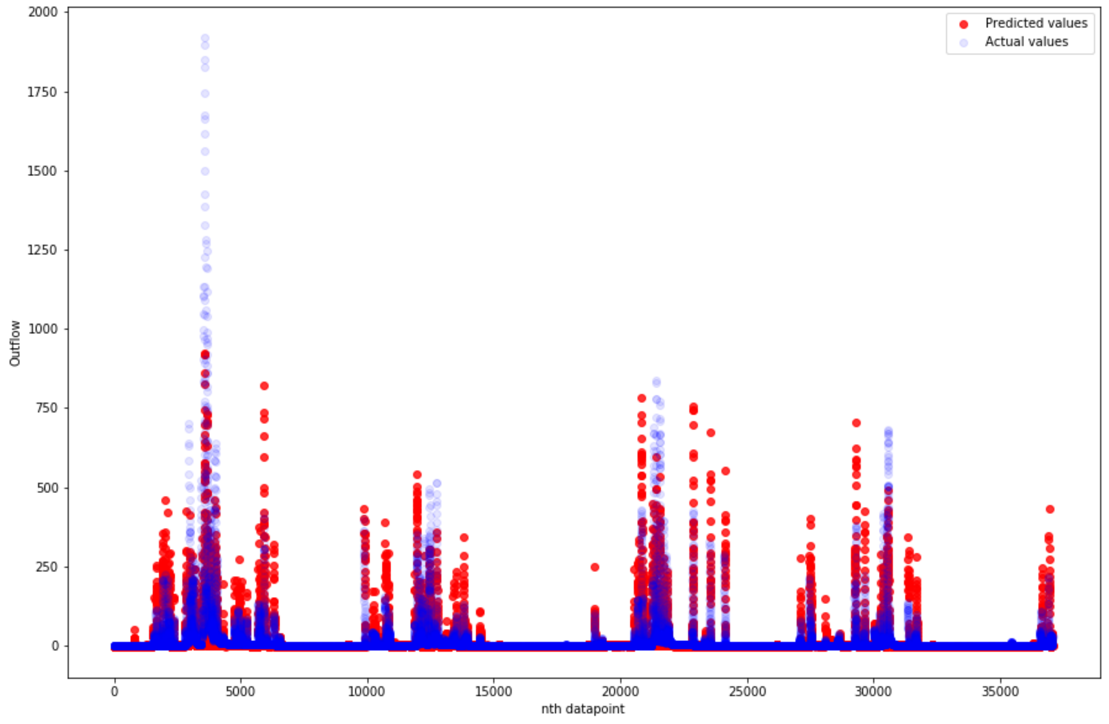
CNN	
LSTM	

Figure 3.4: 1-hour comparisons using Precipitation and Temperature for inputs

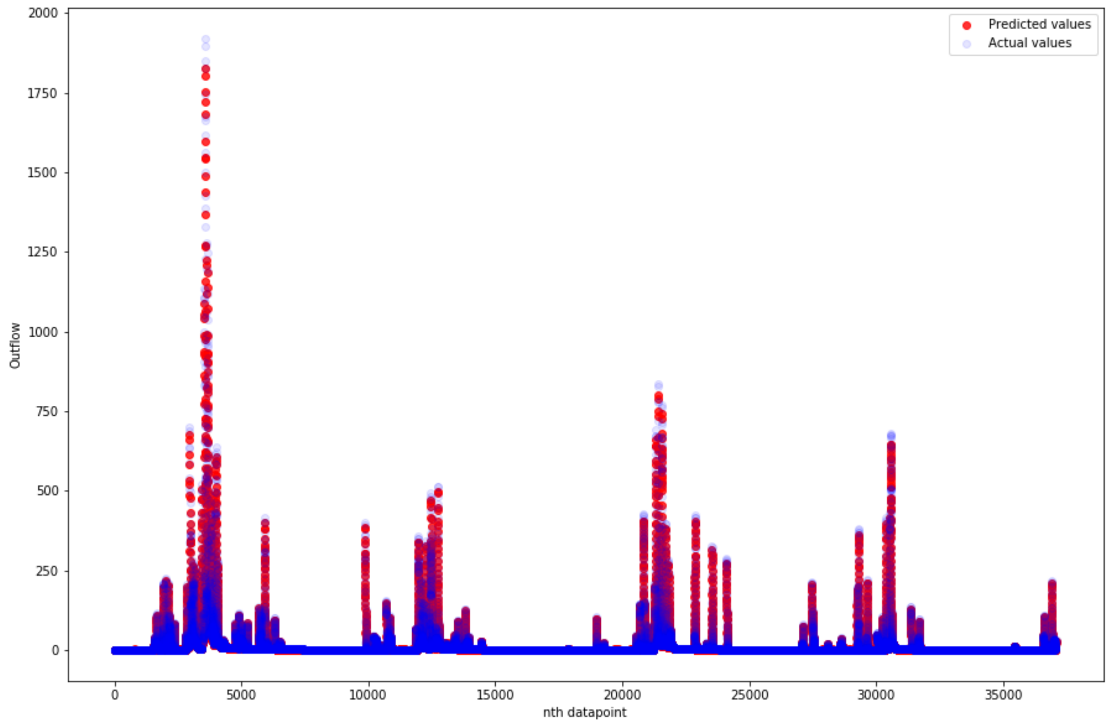
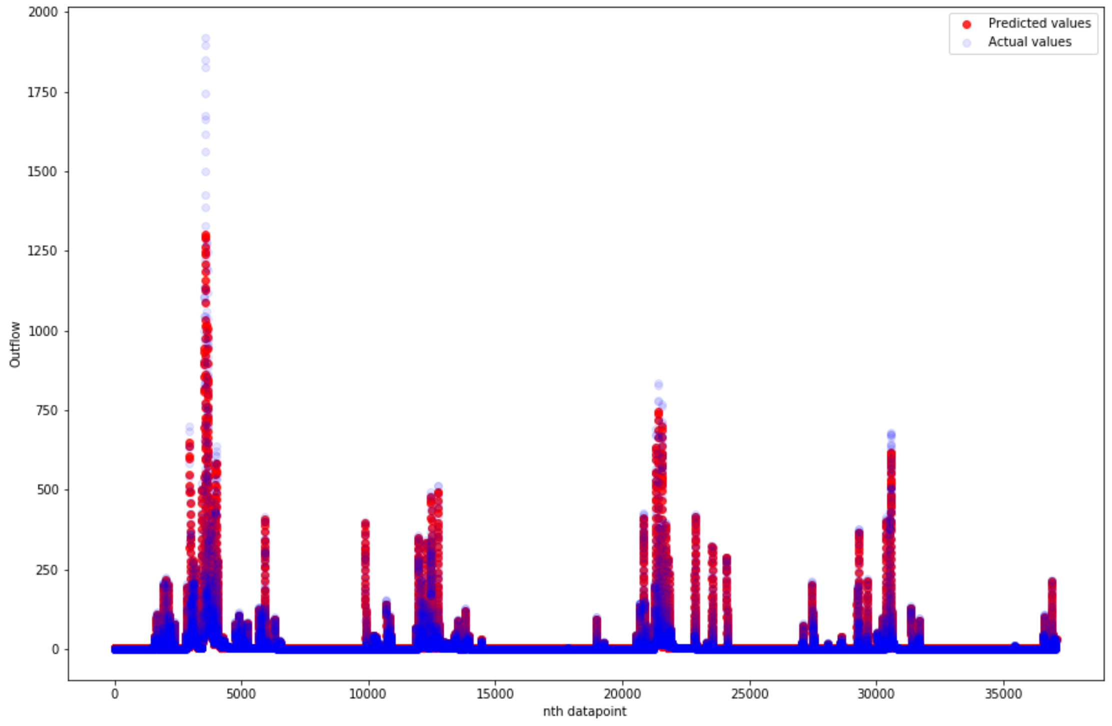
CNN	
LSTM	

Figure 3.5: 1-hour comparisons using Precipitation, Temperature, and Outflow for inputs

3.4.2. <u>2-Hour Predictions</u>

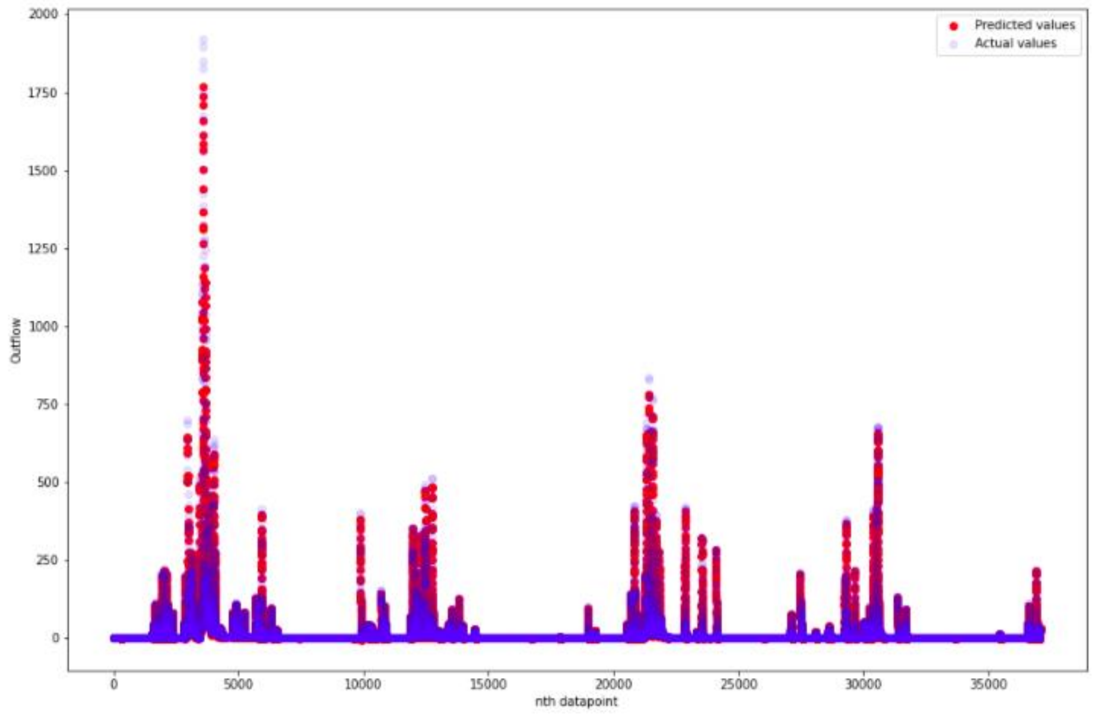
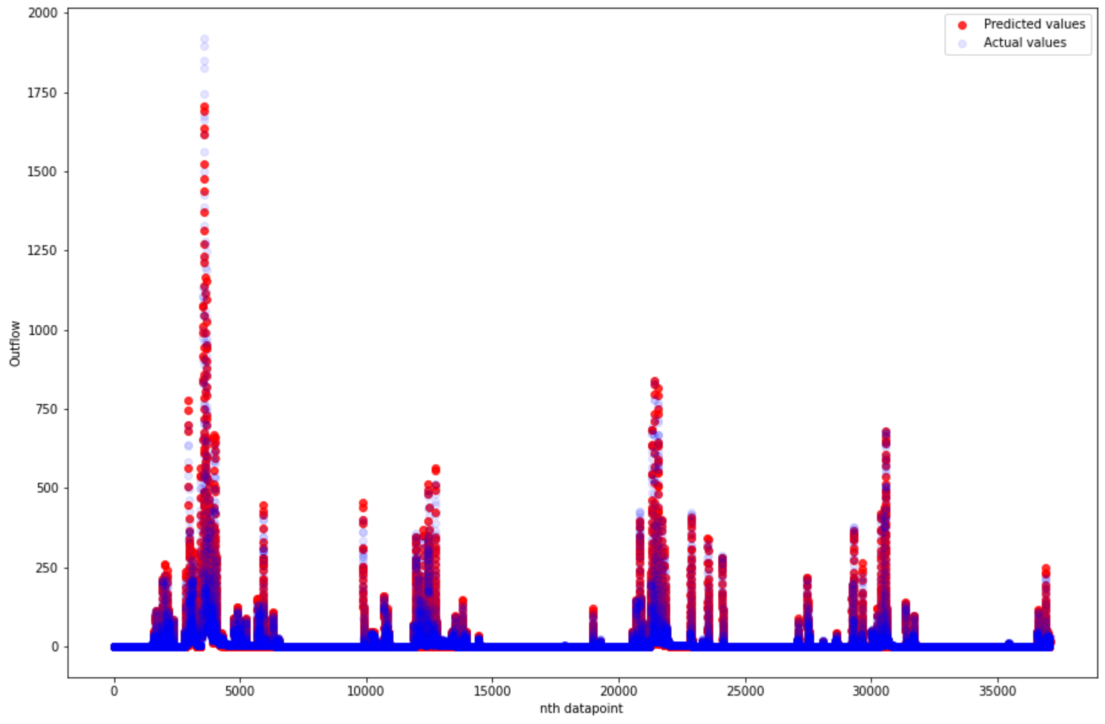
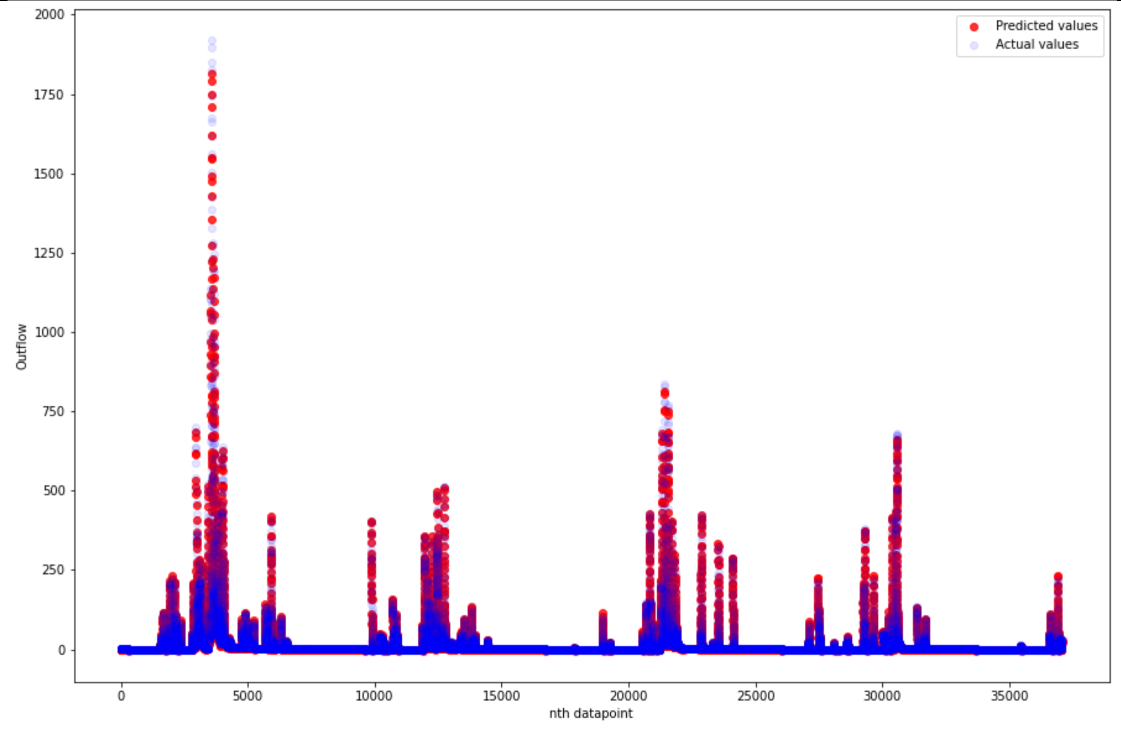
CNN	
LSTM	

Figure 3.6: 2-hour comparisons using Outflow for the input.

CNN



CNN

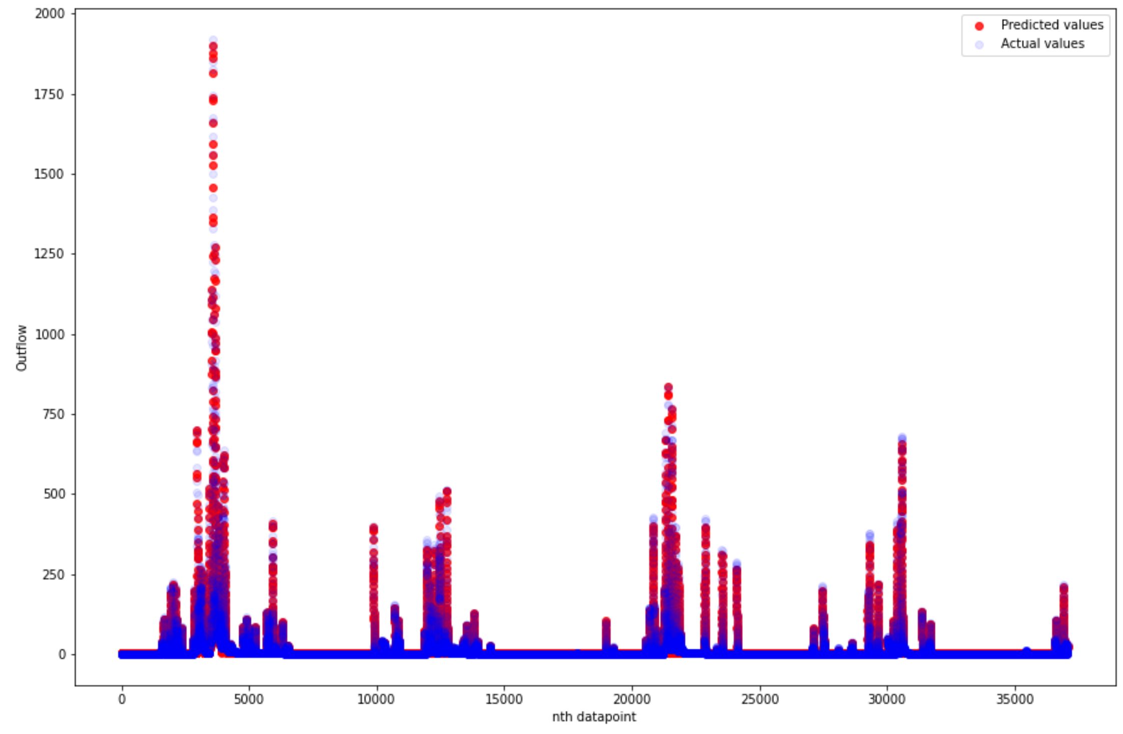


Figure 3.7: 2-hour comparisons using Precipitation and Outflow for inputs

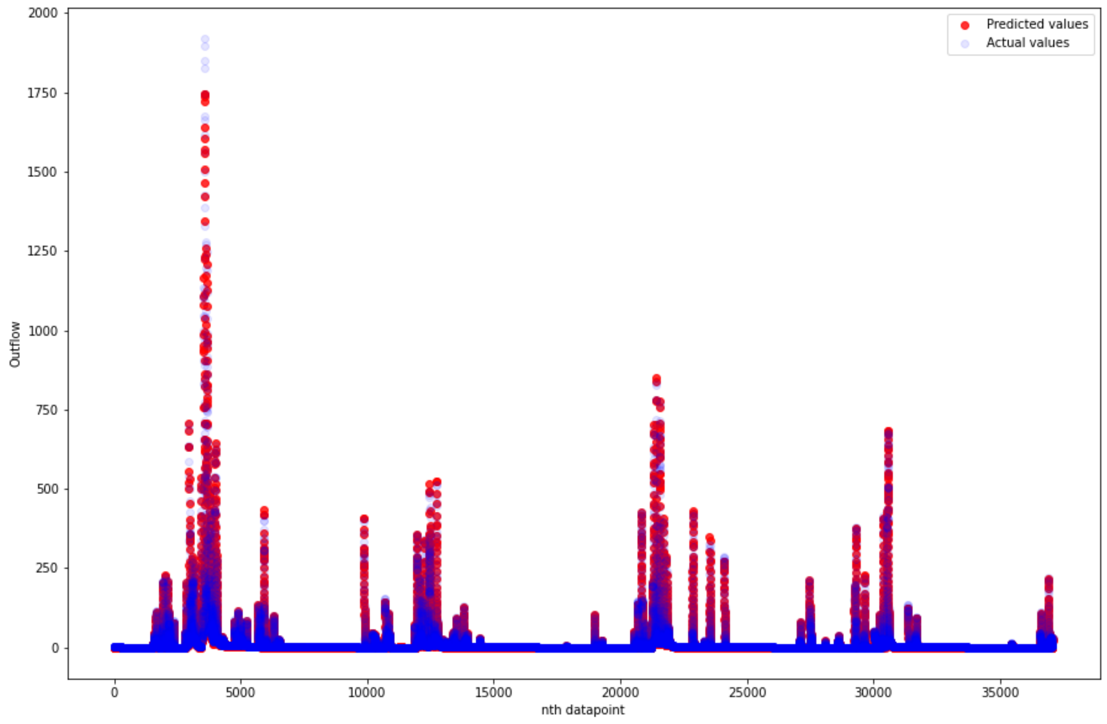
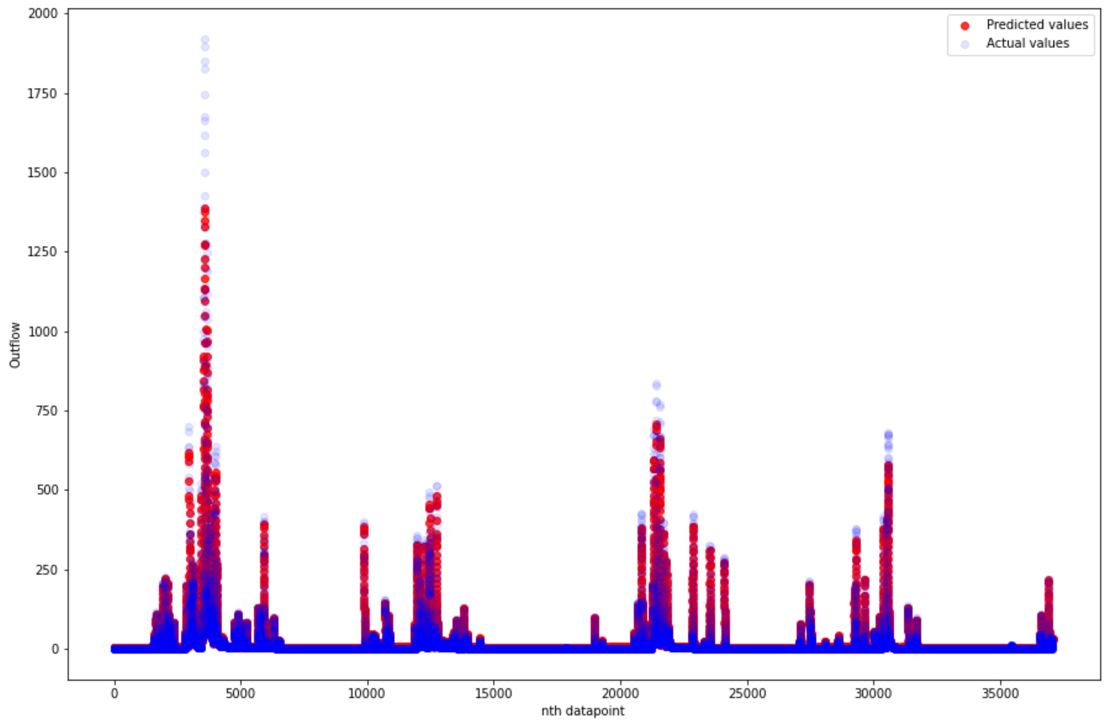
CNN	
LSTM	

Figure 3.8: 2-hour comparisons using Temperature and Outflow for inputs.

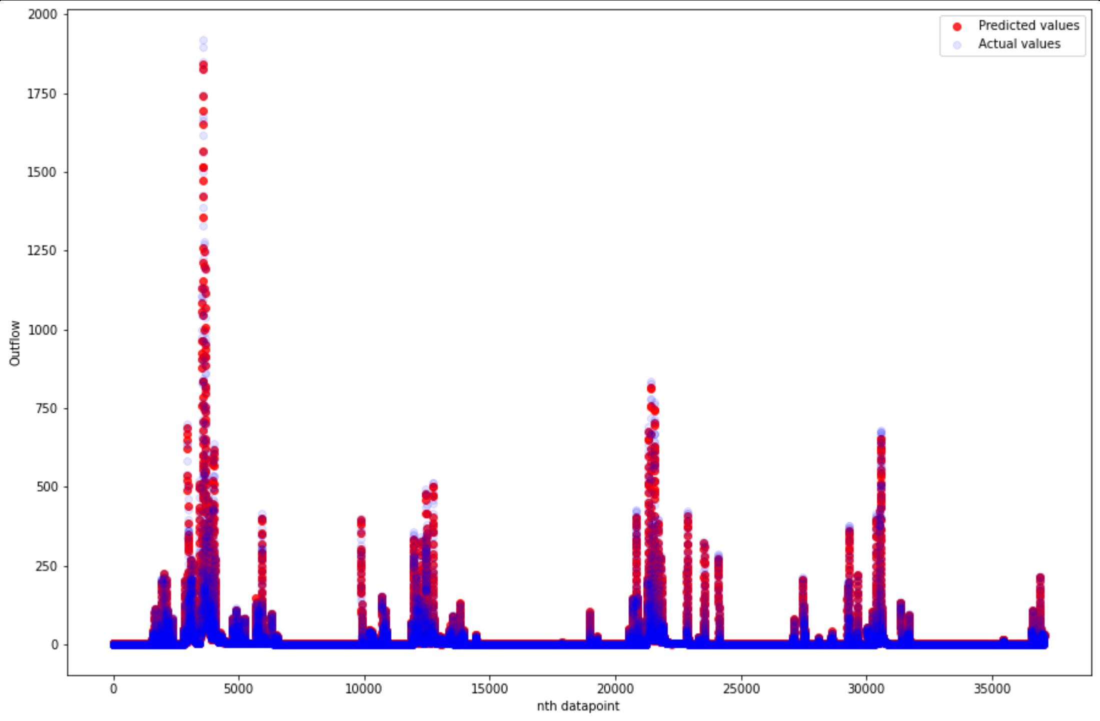
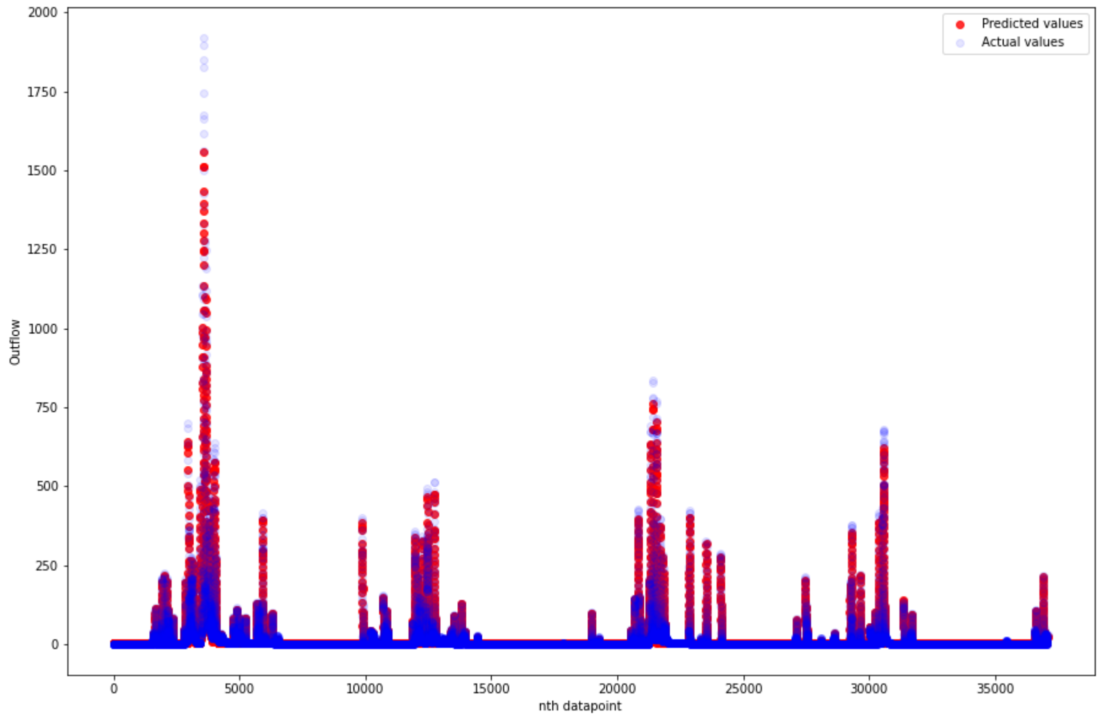
CNN	
LSTM	

Figure 3.9: 2-hour comparisons using Precipitation, Temperature, and Outflow for inputs.

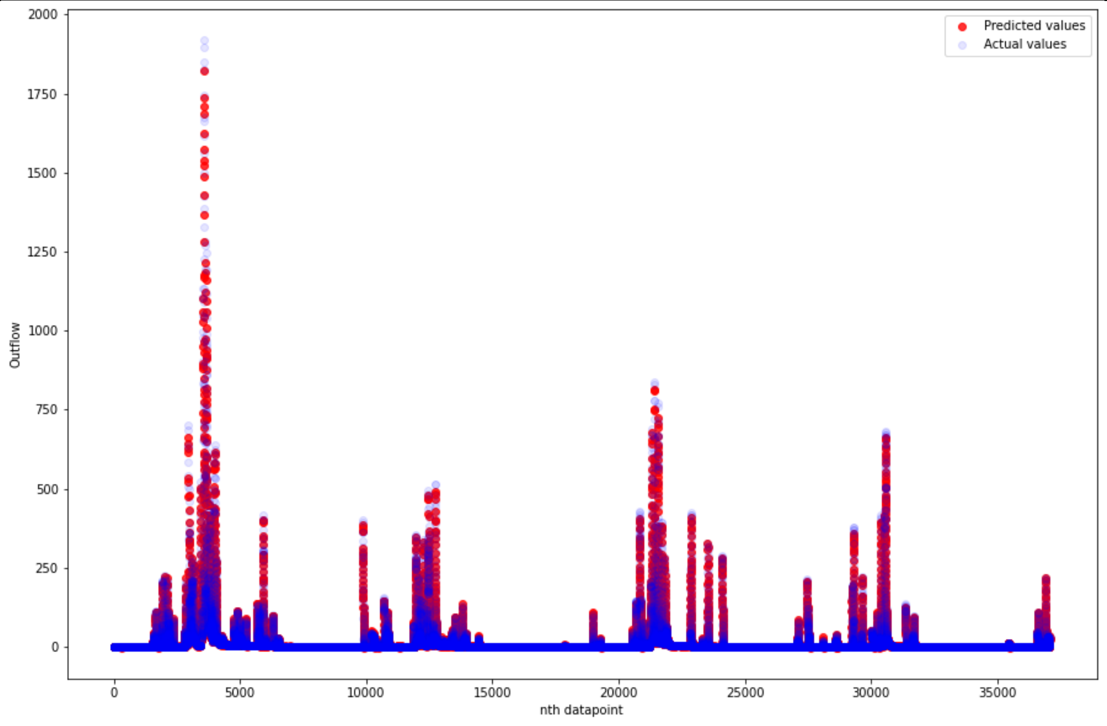
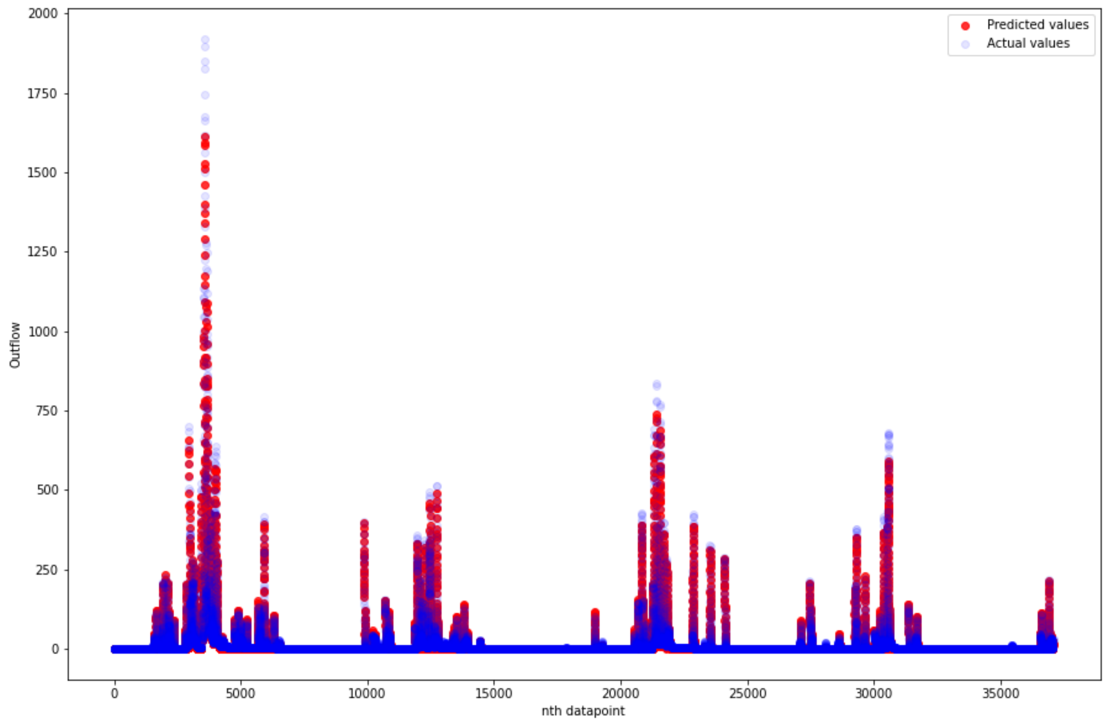
CNN	
LSTM	

Figure 3.10: 2-hour comparisons using Outflow and predicted outflow from hour one

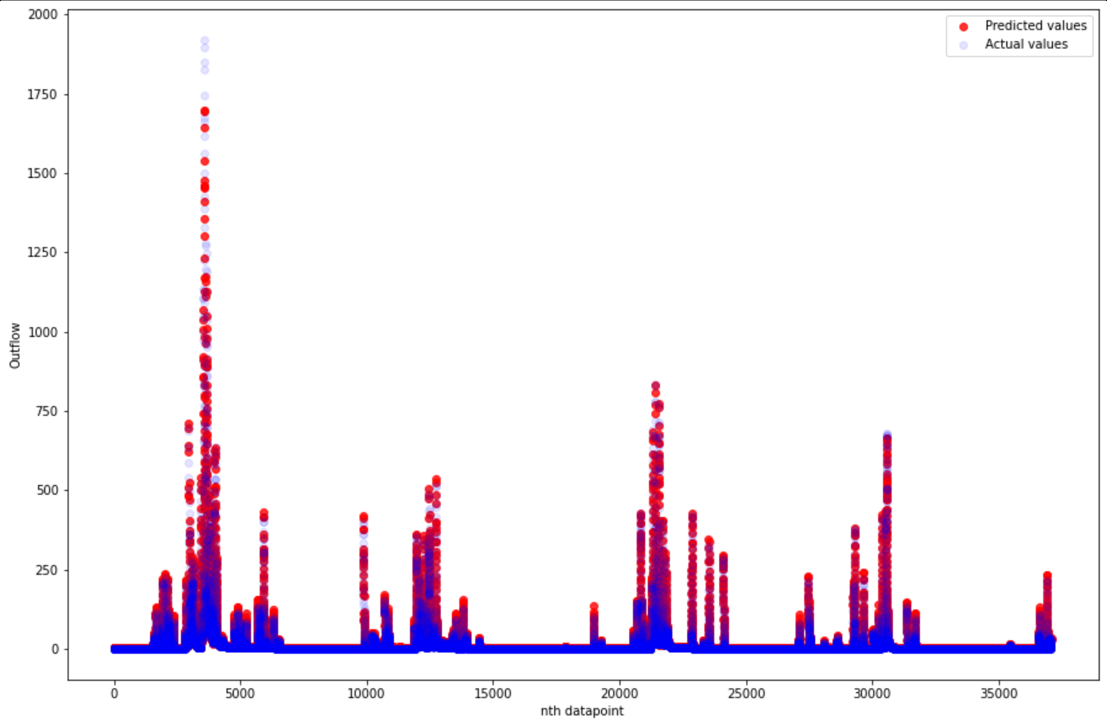
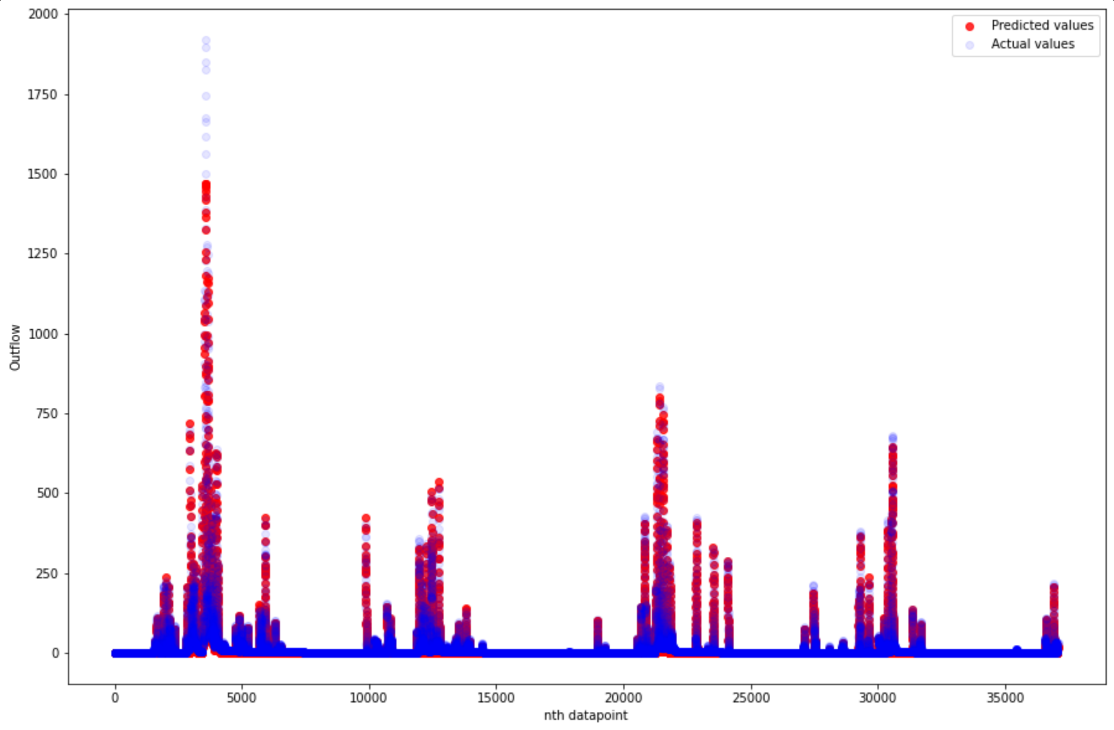
CNN	
LSTM	

Figure 3.11: 2-hour comparisons using Precipitation and Outflow, and predicted precipitation and outflow from hour one

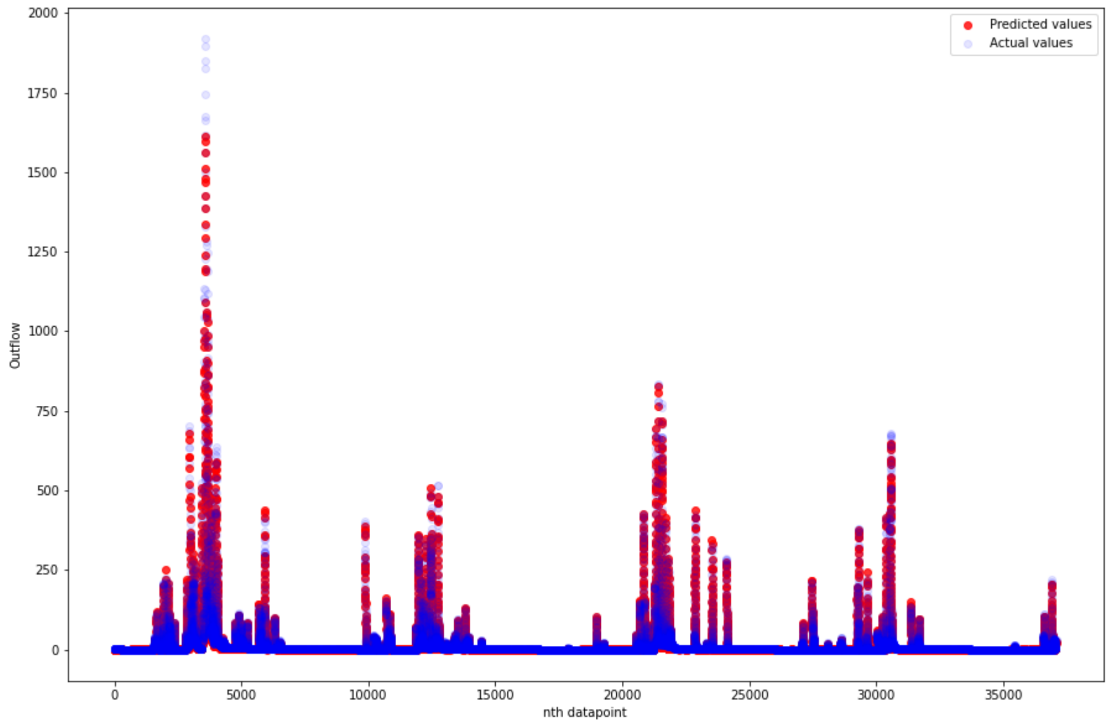
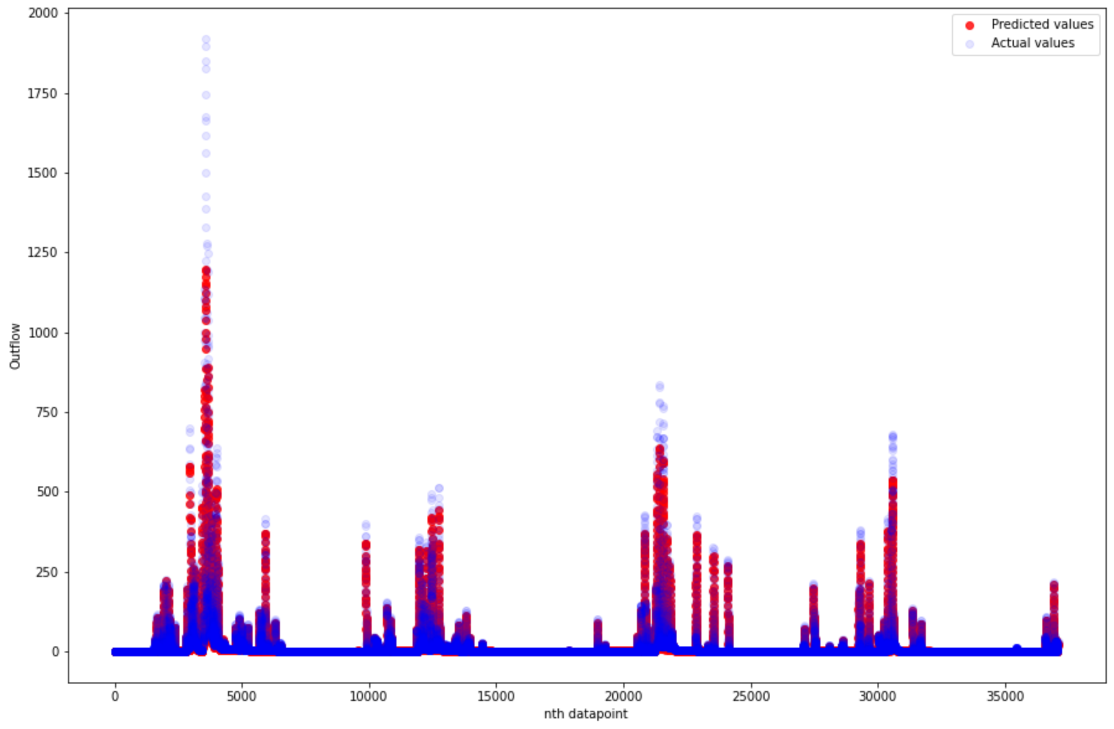
CNN	
LSTM	

Figure 3.12: 2-hour comparisons using Temperature and Outflow, and predicted temperature and outflow from hour one

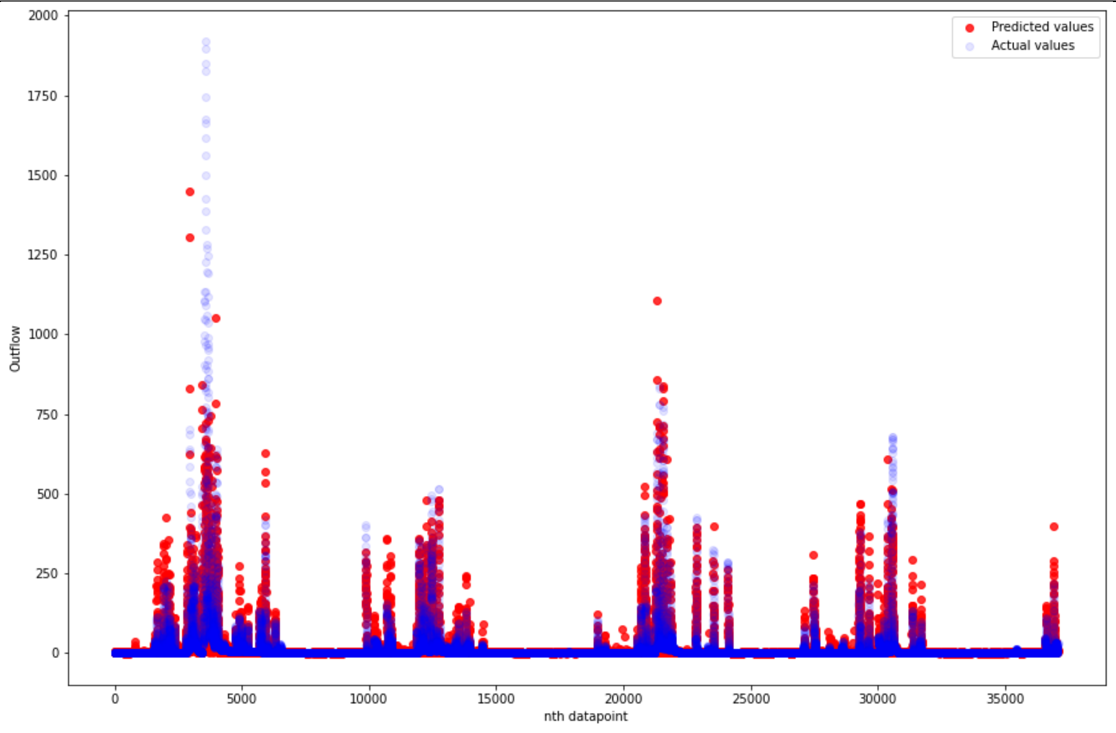
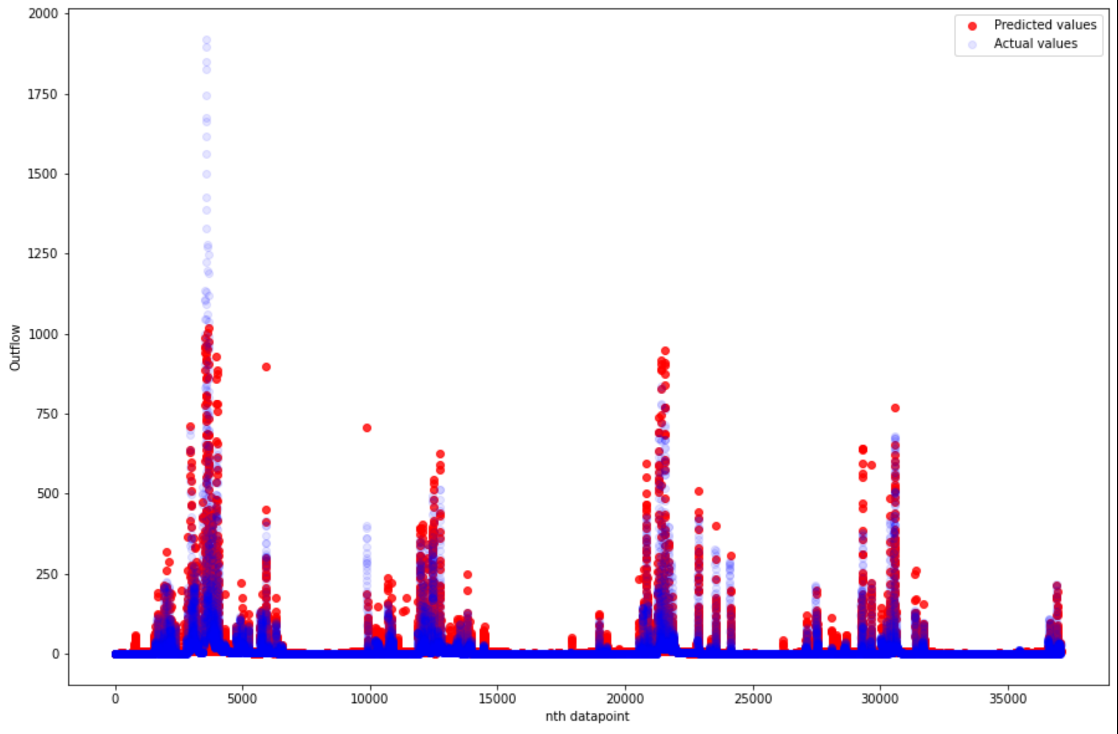
CNN	
LSTM	

Figure 3.13: 2-hour comparisons using Precipitation and Temperature, and predicted precipitation and temperature from hour one

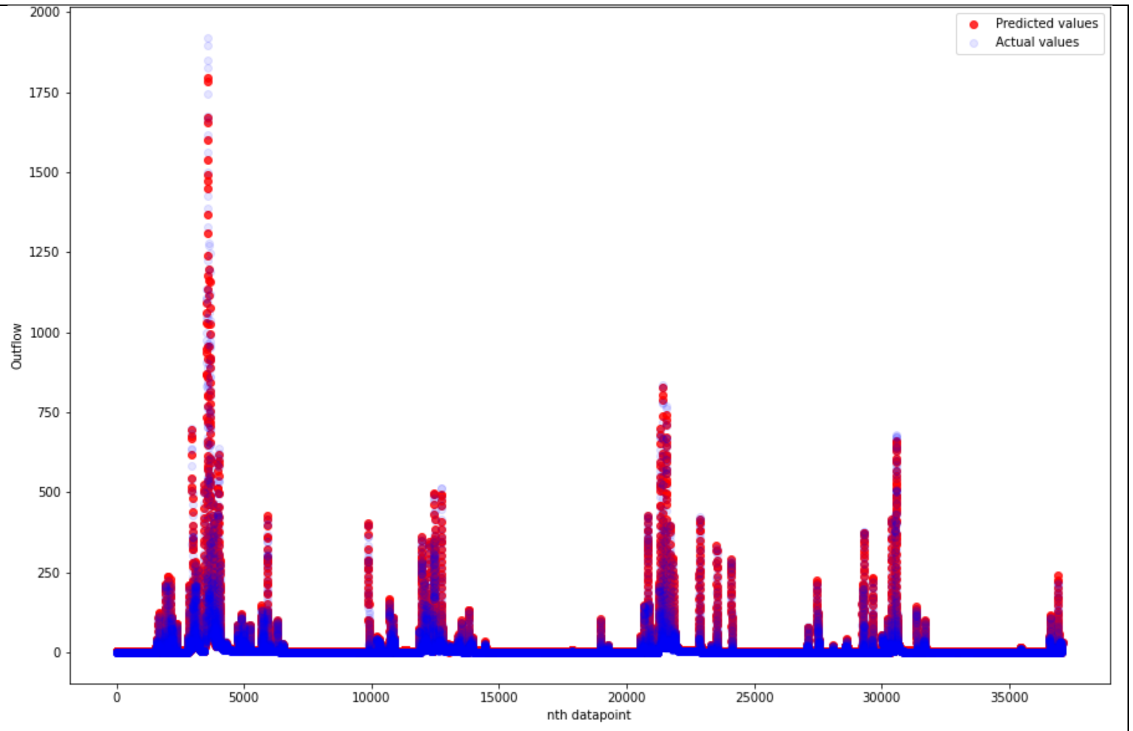
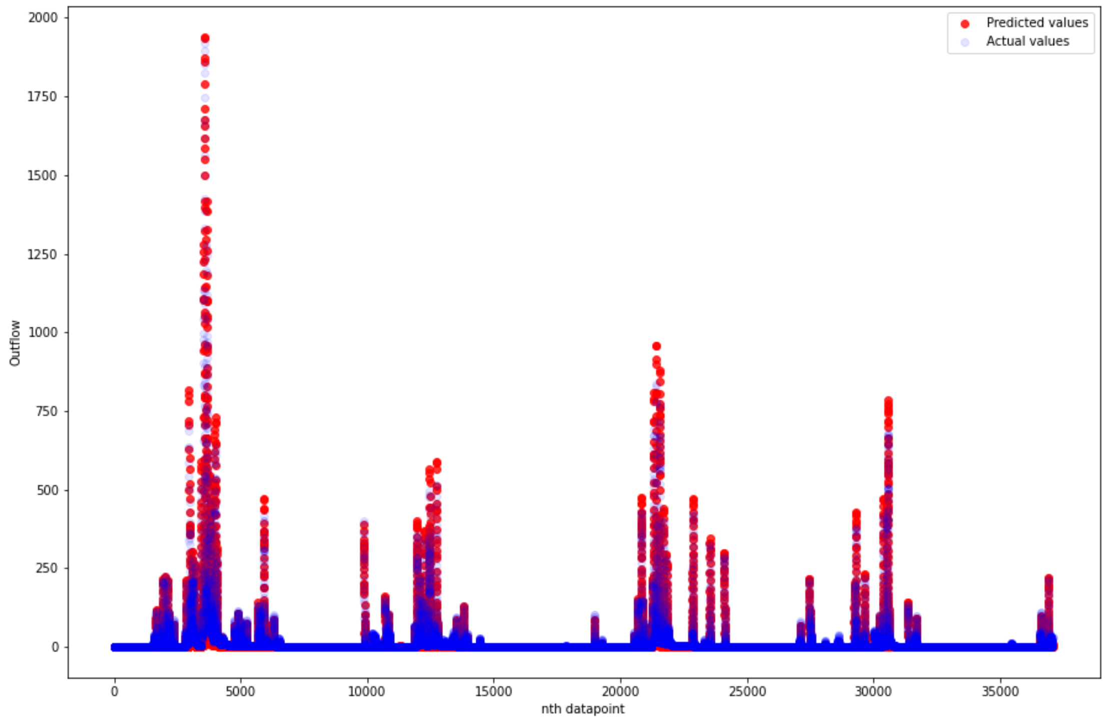
CNN	
LSTM	

Figure 3.14: 2-hour comparisons using Precipitation, Temperature and Outflow, and predicted precipitation, temperature and outflow from hour one

The following attempts were made to improve the performance of the models:

a) inserting the extreme values repeatedly in our original dataset and training the models with this newly injected dataset,
b) training the models with half predicted and half original samples and repeating the training several times with new predictions,
c) training two networks in parallel, one for extreme cases and another for regular data, that are combined by a common feed forward network.
d) training and testing the models only on extreme events.

Unfortunately, none of these methods led to improvements especially for the extreme events.

CHAPTER 4

CONCLUSION

The goal of this thesis was to compare the performances of CNNs and LSTMs for time series predictions. The results from the Dry Creek sub-basin data showed that although CNNs are not specifically designed to predict future observations in time series data, they outperformed the LSTM models which are mainly designed for such prediction problems. The only exception occurred for the models which did not use outflow for the input.   The good performance of the CNN models can be attributed to the fact that through convolution, the models extract highly complex features which couple information across time series predictor variables and across time lags. This study was quite limited in scope because it involved only one time series data set. Therefore, it cannot be concluded that CNNs will, in general, outperform LSTMs in time series forecasting without testing the models on a large ensemble of diverse data sets. Future work will focus on testing the models on additional time series data and also focus on improving the performance of the extreme event prediction models.

REFERENCES

[1] K. Khare, O. Darekar, P. Gupta, and V. Z. Attar, "Short term stock price prediction using deep learning," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2017-05-01 2017: IEEE, doi: 10.1109/rteict.2017.8256643.

[2] M. Hossain, B. Rekabdar, S. J. Louis, and S. Dascalu, "Forecasting the weather of Nevada: A deep learning approach," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015-07-01 2015: IEEE, doi: 10.1109/ijcnn.2015.7280812.

[3] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016-12-01 2016: IEEE, doi: 10.1109/icdm.2016.0061.

[4] P. C. Besse, H. Cardot, and D. B. Stephenson, "Autoregressive Forecasting of Some Functional Climatic Variations," *Scandinavian Journal of Statistics,* vol. 27, no. 4, pp. 673-687, 2000-12-01 2000, doi: 10.1111/1467-9469.00215.

[5] M. Valipour, M. E. Banihabib, and S. M. R. Behbahani, "Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir," *Journal of Hydrology,* vol. 476, pp. 433-441, 2013-01-01 2013, doi: 10.1016/j.jhydrol.2012.11.017.

[6] S. Lee and D. B. Fambro, "Application of Subset Autoregressive Integrated Moving Average Model for Short-Term Freeway Traffic Volume Forecasting," *Transportation Research Record,* vol. 1678, no. 1, pp. 179-188, 1999, doi: 10.3141/1678-22.

[7] R. L. Kashyap, "Optimal Choice of AR and MA Parts in Autoregressive Moving Average Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-4, no. 2, pp. 99-104, 1982-03-01 1982, doi: 10.1109/tpami.1982.4767213.

[8] S. Johansen, "Modelling of cointegration in the vector autoregressive model," *Economic Modelling,* vol. 17, no. 3, pp. 359-373, 2000/08/01/ 2000, doi: https://doi.org/10.1016/S0264-9993(99)00043-7.

[9] I. Houtzager, J.-W. Van Wingerden, and M. Verhaegen, "VARMAX-based closed-loop subspace model identification," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009-12-01 2009: IEEE, doi: 10.1109/cdc.2009.5400695.

[10] F. C. Bagliano and C. A. Favero, "Measuring monetary policy with VAR models: An evaluation," *European Economic Review,* vol. 42, no. 6, pp. 1069-1112, 1998-06-01 1998, doi: 10.1016/s0014-2921(98)00005-1.

[11] L. Yunpeng, H. Di, B. Junpeng, and Q. Yong, "Multi-step Ahead Time Series Forecasting for Different Data Patterns Based on LSTM Recurrent Neural Network," in *2017 14th Web Information Systems and Applications Conference (WISA)*, 2017-11-01 2017: IEEE, doi: 10.1109/wisa.2017.25.

[12] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing,* vol. 323, pp. 203-213, 2019/01/05/ 2019, doi: https://doi.org/10.1016/j.neucom.2018.09.082.

[13] Y. Bengio, P. Frasconi, J. urgen Schmidhuber, and C. Elvezia, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies* Sepp Hochreiter Fakult at f ur Informatik."

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013: PMLR, pp. 1310-1318.

[15] W. K. Li, "Time Series Models Based on Generalized Linear Models: Some Further Results," *Biometrics,* vol. 50, no. 2, p. 506, 1994-06-01 1994, doi: 10.2307/2533393.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997-11-01 1997, doi: 10.1162/neco.1997.9.8.1735.

[17] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation,* vol. 12, no. 10, pp. 2451-2471, 2000-10-01 2000, doi: 10.1162/089976600300015015.

[18] K. Fukushima, "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements," *IEEE Transactions on Systems Science and Cybernetics,* vol. 5, no. 4, pp. 322-333, 1969-01-01 1969, doi: 10.1109/tssc.1969.300225.

[19] D. Scherer, A. Müller, and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in *Artificial Neural Networks – ICANN 2010*: Springer Berlin Heidelberg, 2010, pp. 92-101.

[20] B. Livneh *et al.*, "A spatially comprehensive, hydrometeorological data set for Mexico, the U.S., and Southern Canada 1950–2013," *Scientific Data*, vol. 2, no. 1, p. 150042, 2015-12-01 2015, doi: 10.1038/sdata.2015.42.

[21] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*: Springer, 2012, pp. 9-48.

[22] C. W. J. Granger, "Investigating Causal Relations by Econometric Models and Cross-spectral Methods," *Econometrica,* vol. 37, no. 3, p. 424, 1969-08-01 1969, doi: 10.2307/1912791.

VITA

Graduate School
Southern Illinois University

Bidur Prasad Bhurtel

bidurbhurtel02@gmail.com
bidur.bhurtel@siu.edu

Tribhuvan University, Pulchowk Campus, Nepal
Bachelor of Engineering in Electronics and Communication Engineering, September 2017

Special Honors and Awards:
Master's Fellowship, SIU Carbondale

Thesis Paper Title:
CNNs versus LSTMs for time series forecasting

Major Professor: Dr. Lalit Gupta