Data analysis report paper for time-series traffic data analysis

**1. Introduction**

Traffic volume increases or decreases significantly under the influence of certain factors, and such rapid changes in traffic volume can increase the risk of road traffic or lead to traffic accidents. To prevent this, it is important to predict traffic volume appropriately.

The goal of this project is to analyze the impact of temperature, weather, and time information data on traffic volume.

교통량은 특정 요인의 영향을 받아 크게 상승하거나 감소하는데, 이러한 교통량의 급격한 변화는 도로 교통의 위험성을 증가시키거나 교통 사고 발생으로 이어질 수 있다. 이를 방지하게 위해 적절한 방식으로 교통량을 미리 예측하는 것이 중요하다.

In this study, in order to predict future traffic volume, we compared the performance when predicting with only past traffic volume data as an input and when analyzing it by adding past traffic volume data and other variables such as weather, holidays, and precipitation. When making predictions using a deep learning model, we used three models suitable for predicting time series: LSTM, GRU model, and CNN-LSTM model. After examining all three models, we determined which type of model is more suitable for predicting traffic volume.

본 연구에서는 미래의 교통량을 예측하기 위해 과거의 교통량 데이터를 입력으로 받아 예측을 진행하였을 때와 과거의 교통량과 그 이외 날씨, 연휴, 강수량 같은 변수들을 추가하여 분석을 진행하였을 때의 성능을 비교하였다. 또한 딥러닝 모델을 사용하여 예측을 진행할 때, 시계열을 예측하기에 적합한 LSTM, GRU모델과 CNN-LSTM 모델로 3가지를 사용하고 교통량 예측에 더 적합한 형태의 모델을 결정할 것이다.

As a result of the analysis, the performance was higher and errors were further reduced when only the past traffic volume was in the input than when only past traffic volume data was input. Additionally, it was found that the model with the best performance among LSTM, GRU, and CNN-LSTM models was CNN-LSTM.

분석 결과, 과거의 교통량 데이터만 입력 받을 때보다 교통량 이외의 변수를 입력으로 받았을 때의 성능이 더 높았고 오차 또한 더 감소하는 것으로 나타났다. 또한 LSTM, GRU, CNN-LSTM 모델에서 성능이 가장 좋은 모델은 CNN-LSTM임을 알 수 있었다.

**2. Datasets**

**2.1. Dataset Description**

The dataset we used is the Hourly Minneapolis-St Paul, MN traffic volume for westbound I-94, provided by UCI Machine Learning Repository. It includes a traffic dataset of Hourly Interstate 94 westbound traffic volume for MN DoT ATR station 301, which is roughly midway between Minneapolis and St Paul, MN. As well as the traffic variable, weather, and holiday features from 2012 to 2018 are included for impacts on traffic volume. The traffic data is collected from the Minnesota Department of Transportation and the weather data is provided from Open WeatherMap. The variables in this dataset are as follows:

* holiday: Categorical - US National holidays plus regional holidays, Minnesota State Fair
* temp: Numeric - Average temp in kelvin
* rain\_1h: Numeric - Amount in mm of rain that occurred in the hour
* snow\_1h: Numeric - Amount in mm of snow that occurred in the hour
* clouds\_all: Numeric - Percentage of cloud cover
* weather\_main: Categorical - Short textual description of the current weather
* weather\_description: Categorical - Longer textual description of the current weather
* date\_time: DateTime - Hour of the data collected in local CST time
* traffic\_volume: Numeric - Hourly I-94 ATR 301 reported westbound traffic volume

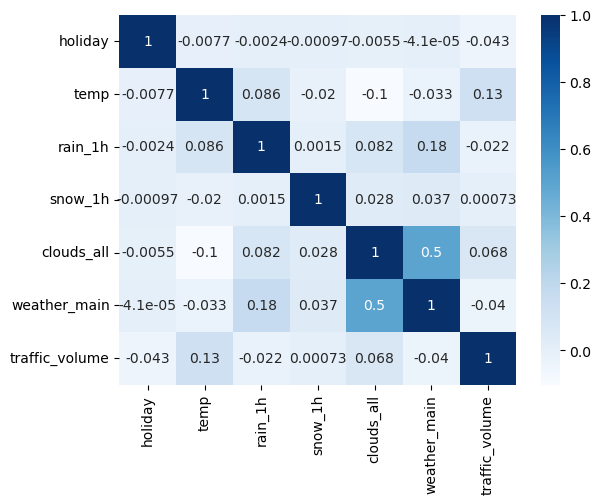
The ‘Holiday’ is a Categorical variable representing US national holidays, regional holidays, and events like the Minnesota State Fair. The ‘temp’ variable is a Numeric variable indicating the average temperature in Kelvin during the specified hour. ‘rain\_1h’ and ‘snow\_1h’ are Numeric variables that mean the amount of rainfall and snowfall in millimeters that occurred in the hour. The ‘clouds\_all’ is a Numeric variable representing the percentage of cloud cover during the specified hour. The ‘weather\_main’ variable is a Categorical variable providing a short textual description of the current weather, indicating the main weather condition. The ‘weather\_description’ variable offers a longer textual description of the current weather. but it was excluded from the analysis. the variable ‘date\_time’ that we used as an index means the hour of data collection in the local Central Standard Time (CST). For the last, the main variable ‘Traffic\_volume’ means the hourly number of the cars crossing this highway I-94.

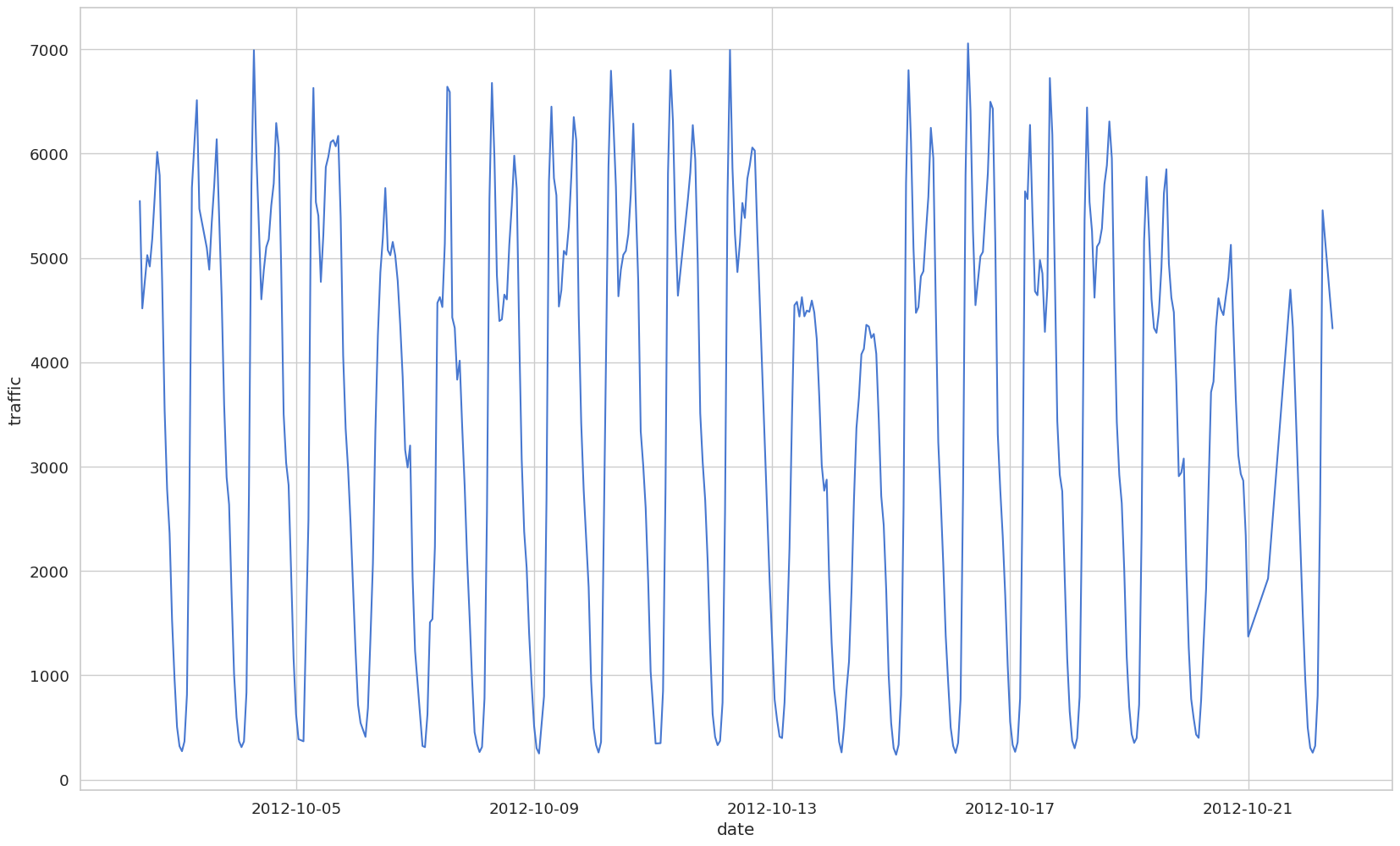
**2.2. Dataset Preprocessing**

In the data preprocessing stage, we made several modifications to the dataset, including altering variable types, removing outliers, and dropping certain variables. Additionally, a linear interpolation method was employed and a sliding window approach was utilized to process the data in 24-hour intervals.

The ‘holiday’ variable was transformed from a categorical form to a binary 0 or 1 representation, and the 'weather\_main' data underwent label encoding. The original dataset presented temperature in Kelvin, which we converted to Celsius for easier interpretation. Additionally, we reformatted the ‘date\_time’ variable to a date type, eliminated outliers in the ‘rain\_1h’ variable, and removed the 'weather\_description' variable.

데이터 전처리는 이상치를 제거하고 결측치를 선형보간 하는 방식으로 사용되었고, 24시간 간격으로 슬라이딩 윈도우 처리되었다. 그 후 라벨인코딩과 원핫인코딩의 과정을 거치고 나머지 변수들을 scaling하는 과정을 진행하였다.

**2.3. Dataset EDA**



**3. The Proposed Approach**

Currently, two prevalent techniques are extensively used for predicting short-term traffic flow because of their superior capability in managing time-based sequences. These methods are Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) (Y. Tao et al, 2020).

**3.1. LSTM**

LSTM is created to manage long-term data dependencies. They have shown effectiveness in utilizing their unique memory cells to recognize long dependencies.

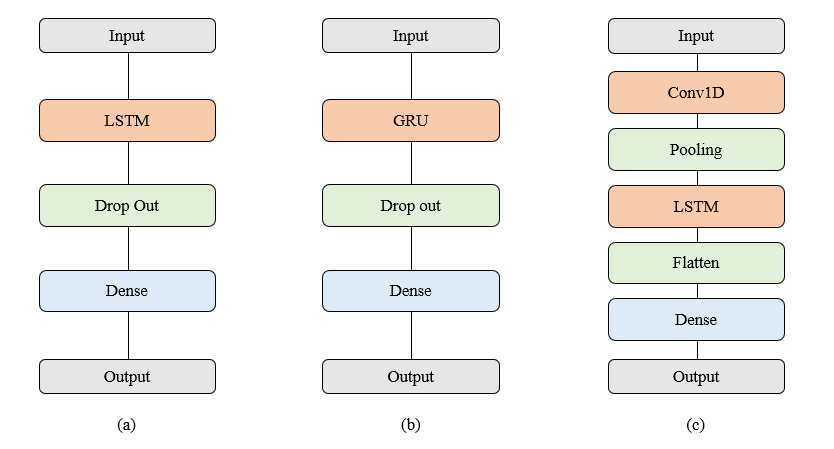
**3.2. GRU**

GRU is slightly different from LSTM, and it was introduced in 2014. Like LSTMs, GRUs handle long-term dependencies and are efficient with sequential data.

**3.3. CNN-LSTM**

To address the challenges encountered when employing LSTM and GRU for short-term traffic flow prediction, particularly in urban road networks with intricate structures and significant travel delays, we have incorporated the use of a Convolutional Neural Network-Long Short-term Memory (CNN-LSTM) model. The CNN-LSTM model enhances the predictive accuracy by effectively capturing these complex spatial-temporal interactions, thereby providing a more robust solution for short-term traffic prediction in challenging urban environments.

This paper evaluates the performance of LSTM, GRU, and CNN-LSTM in traffic volume prediction using a suitable regularization method. The structure of the three models proposed for use in this study can be viewed in Figure 1.



**Figure 1.** Architecture diagrams of (a) LSTM, (b) GRU, and (c) CNN-LSTM.

**4. Evaluation Metrics**

Once the training of the model is finished, it's crucial to assess its performance using the validation set data. Evaluation metrics vary across different tasks, and applying various measurement techniques for comparing the efficacy of distinct models often yields varied outcomes. To effectively examine the regression issue addressed in this paper, metrics such as the mean average absolute error (MAE), mean square error (MSE), and mean absolute percentage error (MAPE) will be utilized as key indicators of performance in the experiment.

모델의 훈련이 끝나면, 검증 세트 데이터를 사용하여 그 성능을 평가하는 것이 중요하다. 평가 지표는 다양한 작업에 따라 다르며, 서로 다른 모델의 효과를 비교하기 위해 다양한 측정 기법을 적용하면 결과도 다양하게 나타난다. 이 논문에서 다루는 회귀 문제를 효과적으로 검토하기 위해, MAE, MSE, MAPE와 같은 지표들을 실험에서 성능의 주요 지표로 사용되었다.

**5. Experimental Analysis**

**5.1. Overall Performances**

To understand the impact of additional variables such as weather, holidays, and precipitation on prediction performance, we conducted modeling in two phases: one using solely traffic volume data and another incorporating these additional variables.

The results of the modeling demonstrated a significant reduction in error rate and a substantial improvement in prediction performance when these additional variables were utilized in conjunction with traffic volume data, as opposed to using traffic volume data alone.

Additionally, the predictive results of the LSTM, GRU, and CNN-LSTM models all showed a high explanatory power of 99%, with the CNN-LSTM model exhibiting the highest performance among them. The specific performances of each model can be found in Table 1 and Table 2.

날씨, 연휴, 강수량의 추가 변수들이 예측 성능에 미치는 영향을 파악하기 위하여, 교통량 데이터만을 활용한 모델과 이러한 추가 변수들을 포함한 모델로 나누어 두 차례에 걸친 모델링을 수행하였다.

모델링 결과, 단순히 교통량 데이터만을 사용했을 때보다 날씨, 연휴, 강수량 등의 추가 변수들을 함께 활용했을 때 오차율이 현저히 감소하며 예측 성능이 크게 향상된 것을 확인할 수 있었다.

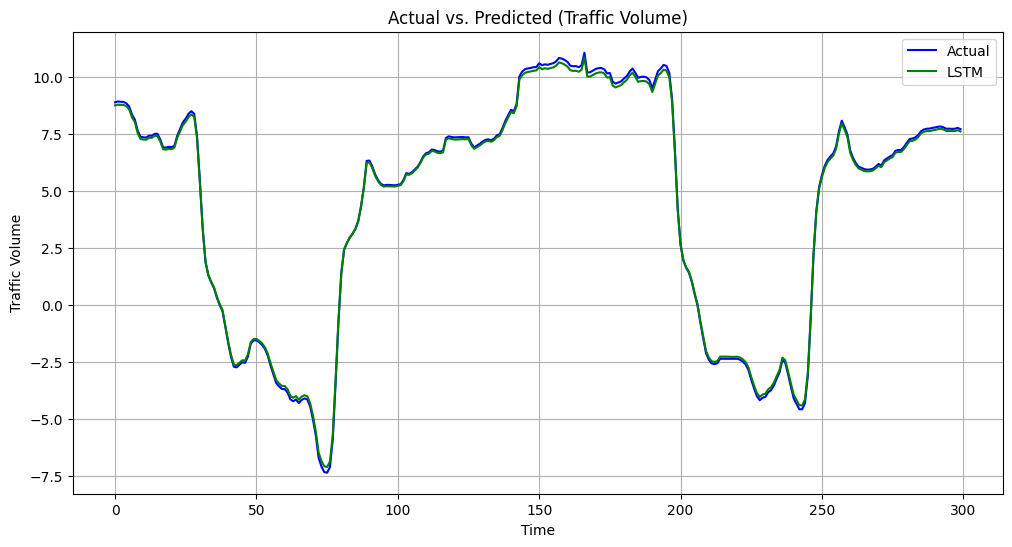
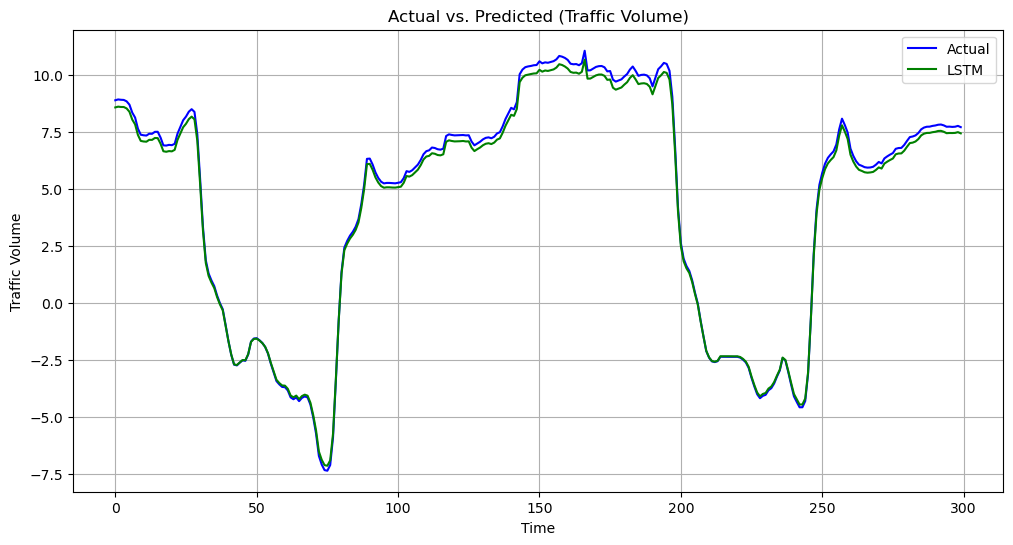
더불어 LSTM, GRU, CNN-LSTM 모델 각각의 예측 결과는 99%의 높은 설명력을 나타내고 있으며, 이 중에서도 CNN-LSTM 모델이 가장 우수한 성능을 보이는 것으로 나타났다.

**Table 1.** Prediction accuracy of train model including only traffic volume variables.

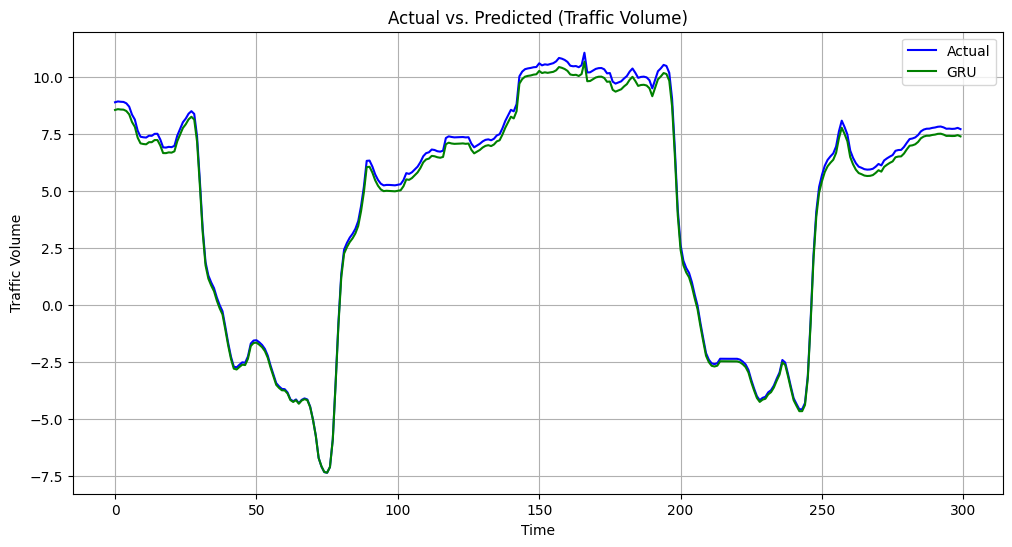
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** |  | **MSE** | **MAE** | **MAPE** |
| **GRU** | 0.9960 | 0.1488 | 0.3426 | 0.1397 |
| **LSTM** | 0.9984 | 0.0578 | 0.2132 | 0.0549 |
| **CNN + LSTM** | 0.9999 | 0.0014 | 0.0347 | 0.0248 |

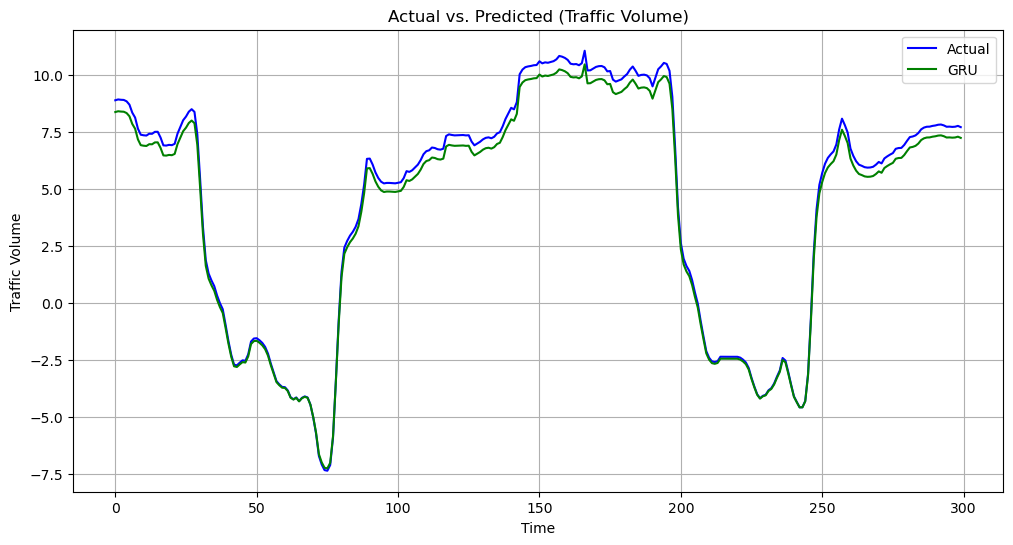
**Table 2.** Prediction accuracy of the train model including the additional variables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** |  | **MSE** | **MAE** | **MAPE** |
| **GRU** | 0.9986 | 0.0529 | 0.2023 | 0.0874 |
| **LSTM** | 0.9993 | 0.0232 | 0.1273 | 0.0402 |
| **CNN + LSTM** | 0.9999 | 0.0014 | 0.0312 | 0.0209 |

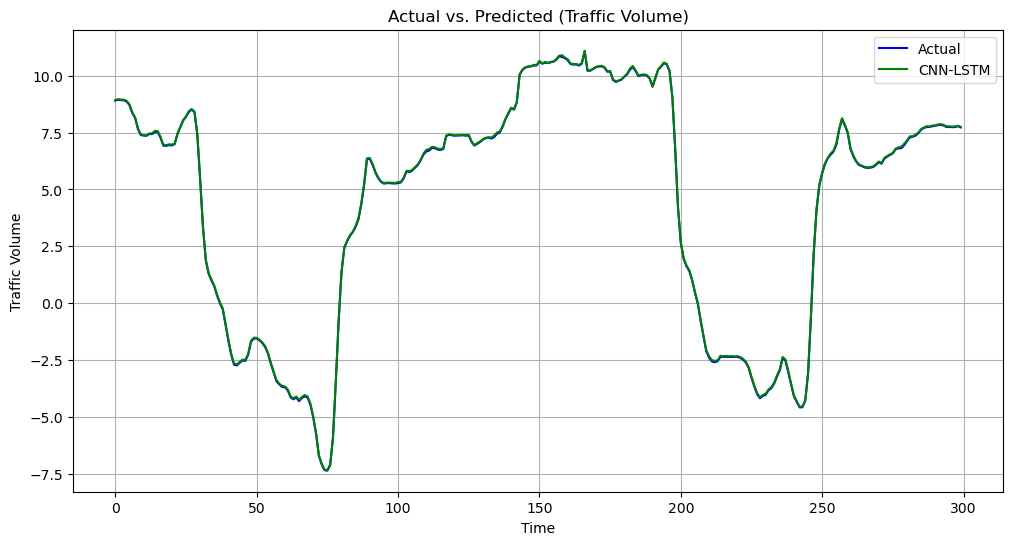
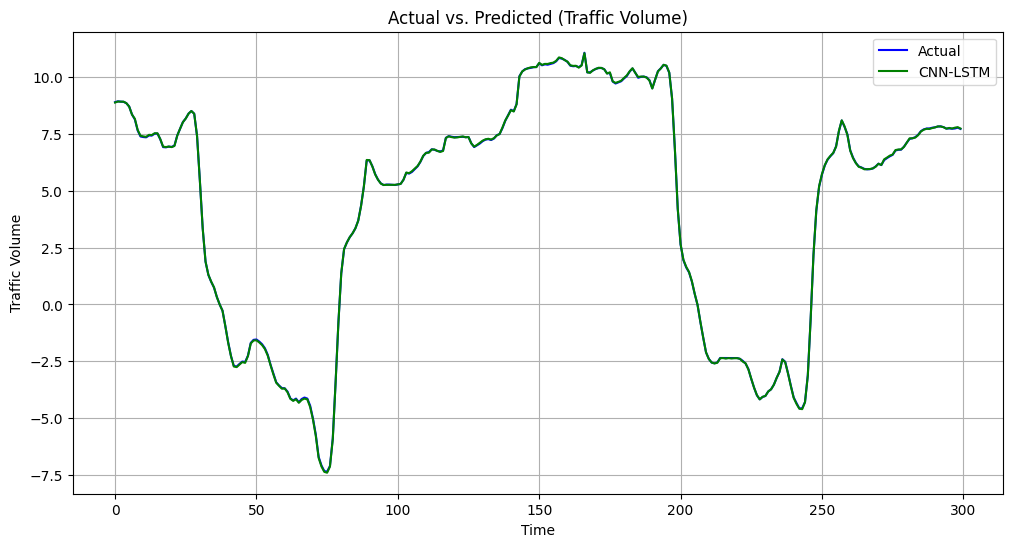


**Figure 2.** Comparison results of the LSTM model excluding and including additional variables.





**Figure 3.** Comparison results of the GRU model excluding and including additional variables.

**Figure 4.** Comparison results of the CNN-LSTM model excluding and including additional variables.

**6. Conclusion**

교통량 변수만을 사용하여 교통량을 예측할 때보다 날씨, 연휴, 강수량 등의 변수를 함께 사용하였을 때, 오차가 크게 감소하였으며,

세 모델의 예측 결과는 모두 99% 이상의 설명력을 보여주고 있고, 그 중 CNN-LSTM의 error(MSE, MAE, MAPE)가 가장 작은 것을 확인할 수 있다.

이번 연구에서는 고속도로에서 수집된 데이터를 사용하였는데, 고속도로의 교통량은 연휴나 강수량 같은 다른 요인들의 영향이 일반 도로 보다 약하기 때문에 추후 연구에서는 고속도로 교통량 이외에도 일반도로의 교통량을 예측하고자 한다. 고속도로의 교통량과 일반도로의 교통량 간의 분포의 차이를 살펴보고 이에 적합한 모델과 분석을 찾아볼 것이다.

When using variables such as weather conditions, holidays, and rainfall in addition to the traffic volume variable for traffic prediction, a significant reduction in error was observed compared to predicting traffic solely based on the traffic volume variable. The predictive results of all three models exhibit an explanatory power of over 99%, with CNN-LSTM showing the smallest errors (MSE, MAE, MAPE) among them.

In this study, data collected from highways was utilized, and the traffic volume on highways is less influenced by factors like holidays or rainfall compared to general roads. Therefore, in future research, we aim to predict traffic volume not only on highways but also on general roads. We will explore the differences in the distribution of traffic volume between highways and general roads and seek suitable models and analyses for these scenarios.

**Reference**

Hogue, John. (2019). Metro Interstate Traffic Volume. UCI Machine Learning Repository. https://doi.org/10.24432/C5X60B.

Minnesota Department of Transportation. Minnesota Department of Transportation - MnDOT. (n.d.). https://www.dot.state.mn.us/

Сurrent weather and forecast - openweathermap. (n.d.). https://openweathermap.org/

P. Poonia and V. K. Jain, "Short-Term Traffic Flow Prediction: Using LSTM," 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Lakshmangarh, India, 2020, pp. 1-4, doi: 10.1109/ICONC345789.2020.9117329.

Ke Wang, Changxi Ma, Yihuan Qiao, Xijin Lu, Weining Hao, Sheng Dong, <A hybrid deep learning model with 1DCNN-LSTM-Attention networks for short-term traffic flow prediction, Volume 583, 2021,126293, ISSN 0378-4371,https://doi.org/10.1016/j.physa.2021.126293.

Y. Tao, P. Sun and A. Boukerche, "A Delay-Based Deep Learning Approach for Urban Traffic Volume Prediction," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149018.

**-코드 structure**(이 문단부터 코드에 대한 자세한 설명과 수식이 이어집니다.)

사용하는 traffic volume 데이터를 불러와서 데이터를 전처리하는 과정을 수행하였다.

<전처리>

이때, 온도를 의미하는 ‘temp’컬럼을 화씨에서 섭씨로 전환하였다.

df['temp']=df['temp']-273.15

또한 연휴를 의미하는 ‘Isholidy’ 컬럼을 연휴이면 ‘1’,연휴가 아니면 ‘0’으로 전환하는 원핫인코딩을 수행하였다.

# holiday를 1 non holiday는 0으로 변환

specific\_holidays = ['Columbus Day', 'Veterans Day', 'Thanksgiving Day', 'Christmas Day', 'New Years Day', 'Washingtons Birthday', 'Memorial Day', 'Independence Day', 'State Fair', 'Labor Day', 'Martin Luther King Jr Day']

df['holiday'] = df['holiday'].apply(lambda x: 1 if x in specific\_holidays else 0)

df.isna().sum() # 결측치 없으므로 결측치 제거는 넘어감

위의 코드를 통해서 결측치를 확인하였고 결측치가 없어 제거과정은 생략하였다.

df['date\_time'] = pd.to\_datetime(df['date\_time'], format='%Y-%m-%d %H:%M:%S')

‘date\_time’컬럼의 형식을 date time의 %Y-%m-%d %H:%M:%S형식으로 전환하였다.

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df['weather\_main'] = encoder.fit\_transform(df['weather\_main'])

또한 시간별 날씨를 한단어로 서술하는 ‘weather\_main’컬럼을 라벨인코딩 수행하여 다수의 범주형 형태로 전환하였다.

# rain 이상치 제거

df = df[df['rain\_1h'] <= 2000]

print(df['rain\_1h'].max())

강수량의 이상치를 확인하고 그 이상치를 제거하기 위해 강수량 2000이상의 데이터는 삭제하였다.

selected\_columns =[ 'holiday', 'temp', 'rain\_1h', 'snow\_1h', 'clouds\_all', 'weather\_main', 'date\_time', 'traffic\_volume']

traffic\_df = df[selected\_columns]

분석에 사용하는 컬럼의 데이터들만 사용하였다.

traffic\_df[traffic\_df.duplicated(['date\_time'], keep = False)]

traffic\_df.drop\_duplicates(inplace=True)

traffic\_df.drop\_duplicates(['date\_time'], inplace=True)

print(len(df)) # raw data: 48,203개

print(len(traffic\_df)) # 중복 삭제 후: 40,574개

위 코드를 통해 ‘date\_time’기준 중복된 데이터를 확인하고 중복된 데이터를 삭제하였다.

중복 삭제 전 48203개의 데이터에서 중복을 삭제하니 40574개의 데이터를 얻었다.

expected\_dates = pd.date\_range(start=traffic\_df['date\_time'].min(), end=traffic\_df['date\_time'].max(), freq='H')

missing\_dates = expected\_dates.difference(traffic\_df['date\_time'])

print("누락된 날짜:")

print(missing\_dates)

len(missing\_dates) # 누락데이터 개수: 11,977개

‘date\_time’컬럼을 기준으로 누락된 데이터를 확인하고 누락된 데이터를 선형보간을 통해 대체하였다.

traffic\_df = traffic\_df.set\_index('date\_time').reindex(expected\_dates)

traffic\_df = traffic\_df.interpolate(method='linear')

traffic\_df = traffic\_df.reset\_index().rename(columns={'index': 'date\_time'})

print(traffic\_df.isna().sum())

print(traffic\_df.info()) # 최종 데이터 개수: 52,551개

전처리된 데이터를 마지막으로 다시 저장해두고, 이 데이터로 EDA를 수행한다.

#train\_test분리

def data\_split(df):

n = len(df)

train\_df = df[0:int(n\*0.8)]#8할이 train

test\_df = df[int(n\*0.8):] #2할이 test

c = df.shape[1]

return train\_df, test\_df, c

그 후, train test set을 분할하는 함수를 정의해 데이터 분리에 사용한다.

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() #스케일링

traffic\_df= pd.DataFrame(scaler.fit\_transform(traffic\_df))

y= pd.DataFrame(scaler.fit\_transform(y))

x\_train, x\_test, x\_nf=data\_split(traffic\_df) #train test set split

y\_train, y\_test, y\_nf=data\_split(y)

StandardScaler를 통해 변수 스케일링을 진행하고 위에서 정의한 data\_split함수로 데이터셋 분리를 수행한다. (train데이터가 8할, test데이터는 2할의 비율이다)

WINDOW\_SIZE=24 # 24시간 기준

x\_train\_window = x\_window\_re(x\_train, WINDOW\_SIZE)

y\_train\_window = y\_window\_re(y\_train, WINDOW\_SIZE)

x\_test\_window = x\_window\_re(x\_test, WINDOW\_SIZE)

y\_test\_window = y\_window\_re(y\_test, WINDOW\_SIZE)

그후, 24시간을 기준으로 슬라이딩 윈도우 처리를 거친 데이터를 반환 받는다.

이제 전처리는 모두 진행되었고, 아래부터는 모델링 파트이다.

<모델링>

LEARNING\_RATE = 0.01

BATCH\_SIZE = 256

EPOCHS = 10

PCA\_COMPONENT = 4

WINDOW\_SIZE=24

TRAINDATA = '교통량\_train\_windowsize\_24.npz'

TESTDATA = '교통량\_test\_windowsize\_24.npz'

window\_size =WINDOW\_SIZE

import numpy as np

모델의 학습에 사용하기 위한 하이퍼파라미터는 다음과 같다.

-LSTM 모델

from tensorflow import keras

import matplotlib.pyplot as plt

from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error

model = keras.Sequential()

model.add(keras.layers.LSTM(10, activation='relu', return\_sequences=True, input\_shape=(24, x\_nf)))

model.add(keras.layers.Dropout(0.1))

# Dense 레이어 수정

model.add(keras.layers.Dense(1, activation='linear')) # 'linear' 활성화 함수 사용

model.summary()

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['mae'])

history=model.fit(x\_train\_window, y\_train\_window, validation\_data=(x\_test\_window,y\_test\_window), epochs=200)

# 모델 평가

y\_pred1 = model.predict(x\_test\_window)

# R^2

r2 = r2\_score(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1)) # 열기준 합을 비교

print("R^2:", r2)

# MSE

mse = mean\_squared\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MSE:", mse)

# MAE

mae = mean\_absolute\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAE:", mae)

#MAPE

mape = mean\_absolute\_percentage\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAPE:", mape)

loss\_history1 = history.history['loss']

plt.figure(figsize=(12, 6))

plt.plot(loss\_history1, label='loss', linestyle='-', marker='', color='blue')

plt.show()

그 후, 모델의 예측이 제대로 이루어지고 있는지 확인하기 위해 실제값과 예측값의 그래프를 출력하는 코드를 사용하여 모델의 성능을 확인한다.

import matplotlib.pyplot as plt

# 실제값

actual\_values = y\_test\_window.sum(axis=1)

import numpy as np

actual\_values = np.array(actual\_values)

# 예측값

pred1 = y\_pred1.sum(axis=1)

# 시간 단계 생성 (예: 1, 2, 3, ...)

time\_steps = range(len(actual\_values))

# 하나의 시간 창 선택 (예: 0번째 시간 창)

time\_window\_index = 1

# traffic\_volume 데이터 선택

traffic\_volume\_actual = actual\_values[:, 0]

traffic\_volume\_pred1 = pred1[:, 0]

# 그래프 그리기

plt.figure(figsize=(12, 6))

plt.plot(time\_steps[:300], traffic\_volume\_actual[:300], label='Actual', linestyle='-', marker='', color='blue') # 실선, 마커 없음

plt.plot(time\_steps[:300], traffic\_volume\_pred1[:300], label='LSTM', linestyle='-', marker='', color='green') # 점선, 마커 없음

plt.title('Actual vs. Predicted (Traffic Volume)')

plt.xlabel('Time')

plt.ylabel('Traffic Volume')

plt.legend()

plt.grid(True)

plt.show()

-GRU 모델

실제와 예측값을 출력하는 그래프 코드는 동일하므로 첨부하지 않고 모델의 실행 코드만 첨부하였다.(실제 코드엔 모두 존재)

from tensorflow import keras

import matplotlib.pyplot as plt

from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error

model = keras.Sequential()

model = keras.Sequential()

model.add(keras.layers.GRU(10, activation='relu', return\_sequences=True,input\_shape=(24, x\_nf)))

model.add(keras.layers.Dropout(0.1))

model.add(keras.layers.Dense(1, activation='linear')) # 'linear' 활성화 함수 사용

model.summary()

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['mae']) # MSE 손실 함수 사용

history=model.fit(x\_train\_window, y\_train\_window, validation\_data=(x\_test\_window,y\_test\_window), epochs=200)

# 모델 평가

y\_pred1 = model.predict(x\_test\_window)

# R^2

r2 = r2\_score(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))#열기준 합을 비교

print("R^2:", r2)

# MSE

mse = mean\_squared\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MSE:", mse)

# MAE

mae = mean\_absolute\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAE:", mae)

loss\_history1 = history.history['loss']

#MAPE

mape = mean\_absolute\_percentage\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAPE:", mape)

plt.figure(figsize=(12, 6))

plt.plot(loss\_history1, label='loss', linestyle='-', marker='',color='blue')

-CNN-LSTM 모델

CNN-LSTM모델 또한 중복되는 부분은 제외하고 모델의 실행 코드만 아래 작성하였다.

from tensorflow import keras

from tensorflow.keras.layers import Input, Conv1D, LSTM, Dense, concatenate, Embedding, Flatten, GRU

import numpy as np

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, LSTM, Dense

import tensorflow\_addons as tfa

model = keras.Sequential()

model.add(keras.layers.Conv1D(32, 3, activation='relu', input\_shape=(24, x\_nf)))

model.add(MaxPooling1D(2))

model.add(LSTM(32, activation='relu', return\_sequences=True))

model.add(Flatten())

model.add(Dense(24, activation='linear')) # Output shape set to 1

model.summary()

early\_stop = keras.callbacks.EarlyStopping(patience=1, restore\_best\_weights=True)

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics='mae')

history = model.fit(x\_train\_window, y\_train\_window, validation\_data=(x\_test\_window, y\_test\_window), epochs=200)

# 모델 평가

y\_pred1 = model.predict(x\_test\_window)

# R^2

r2 = r2\_score(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1)) # 열기준 합을 비교

print("R^2:", r2)

# MSE

mse = mean\_squared\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MSE:", mse)

# MAE

mae = mean\_absolute\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAE:", mae)

#MAPE

mape = mean\_absolute\_percentage\_error(y\_test\_window.sum(axis=1), y\_pred1.sum(axis=1))

print("MAPE:", mape)

loss\_history1 = history.history['loss']

plt.figure(figsize=(12, 6))

plt.plot(loss\_history1, label='loss', linestyle='-', marker='', color='blue')

plt.show()

이 코드들을 통해 모델의 예측과 실제값의 그래프를 화면에 출력하며 모델의 성능을 평가하는 지표를 확인하게 된다.