KiSTi 한국과학기술정보연구원 Korea Institute of Science and Technology Information

2022

Advanced Topic in Research Data-centric Deep Learning

Lec 10: Seq2Seq and Attention

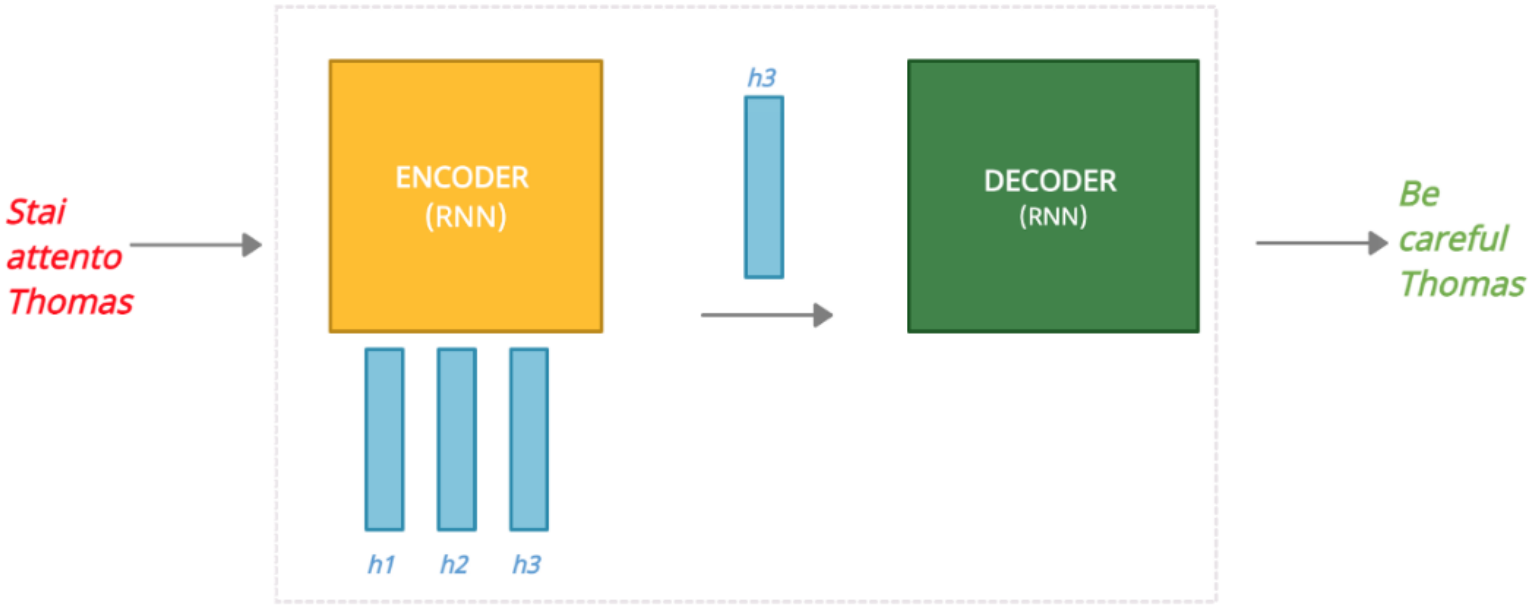
hsyi@kisti.re.kr

Hongsuk Yi (이홍석)

Reviewing the last class: Attention

Limitation of Seq2Seq model

- Main problem with seq2seq models
  - ✓compress all the information into one fixed-size vector results in information loss.
- This is the problem that attention solves!
  - ✓The last **hidden state (*h3*)** becomes the content that is sent to the decoder
  - ✓the encoder is “forced” to send only a single vector, regardless of the length of our input



Attention: a general formulation

- Attention weights, we apply the softmax of the “importance score”
- The “summary” is computed as a weighted sum

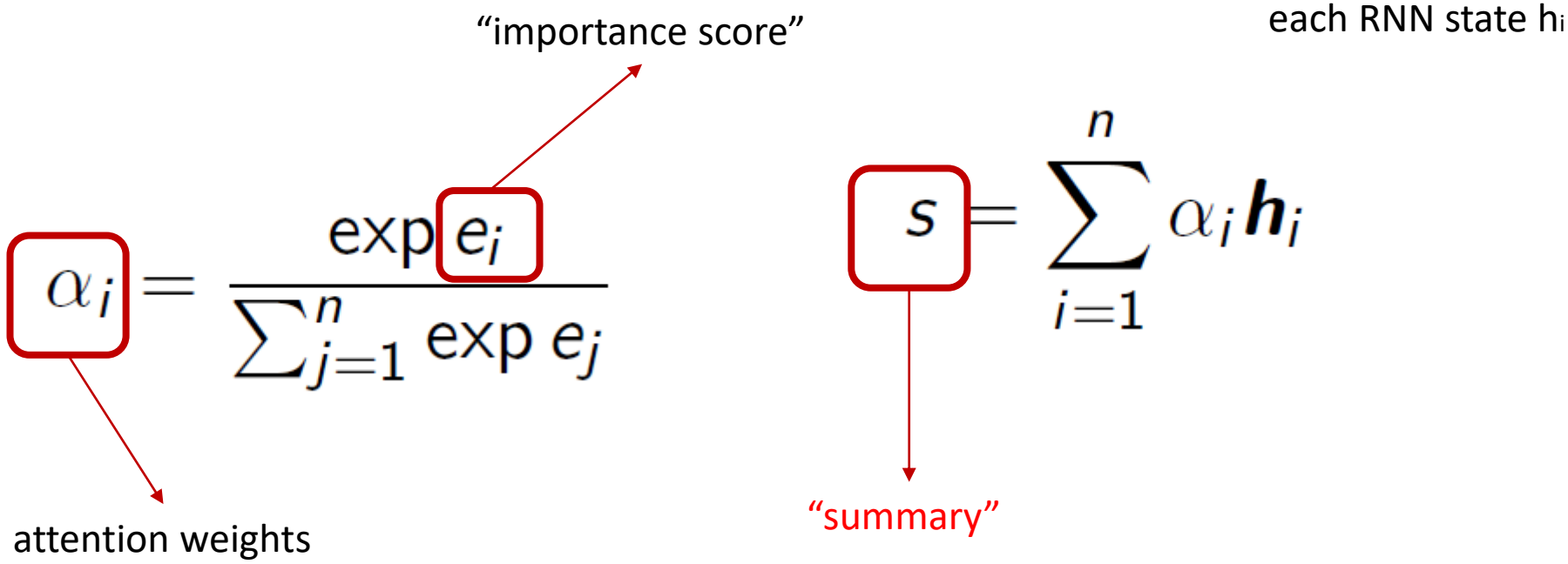


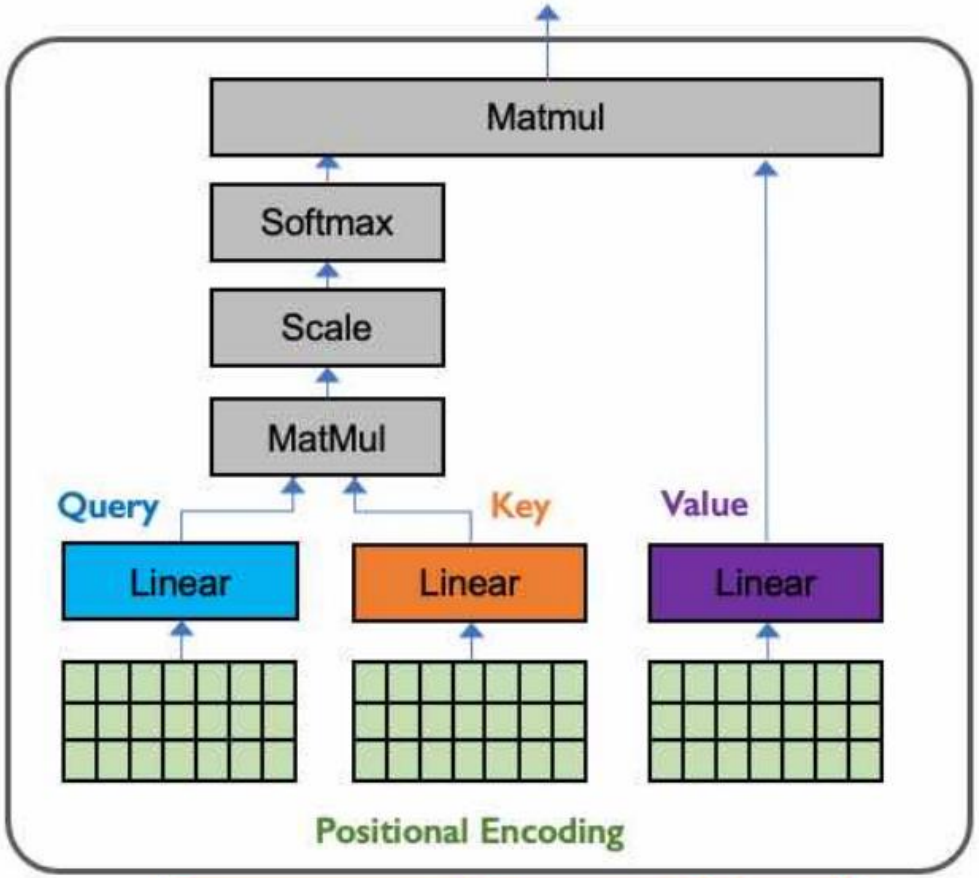
image borrowed from Richard Johansson (*Chalmers Technical University and University of Gothenburg)*

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

These operations form a self-attention head that can plug into a larger network. Each head attends to a different part of input.



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$

Today:
Transformer model for Traffic Flow Prediction

Attention: a general formulation

- What is “importance score” for each state (h)

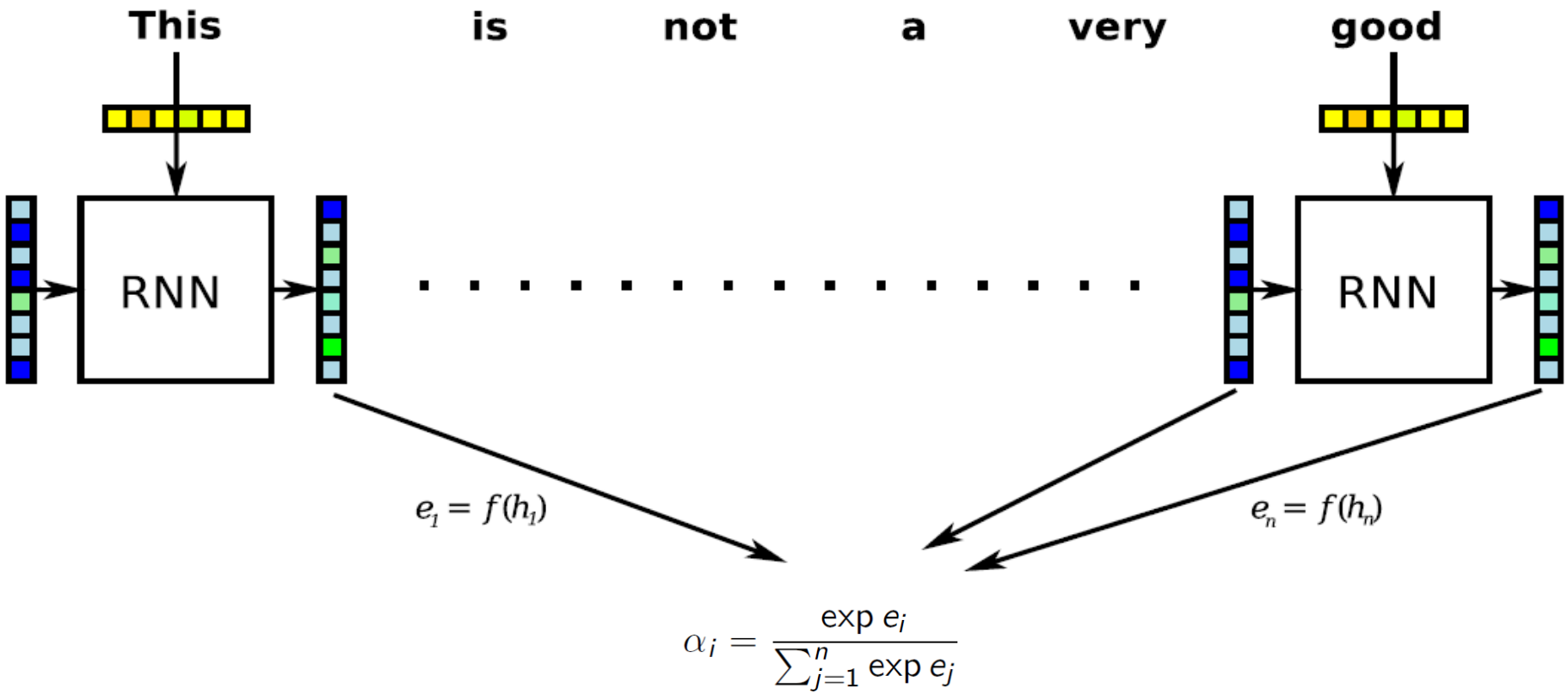


image borrowed from Richard Johansson (*Chalmers Technical University and University of Gothenburg)*

Attention: a general formulation

- Attention weights, we apply the softmax of the "importance score"
- The "summary" is computed as a weighted sum

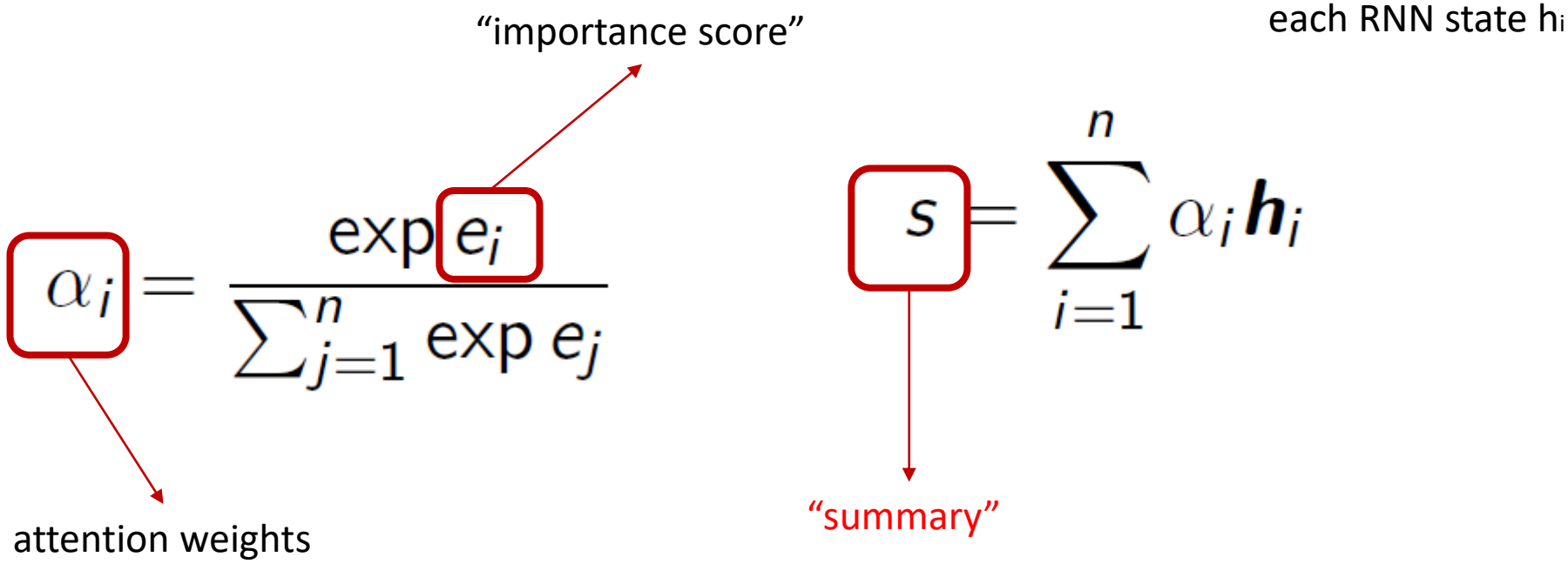


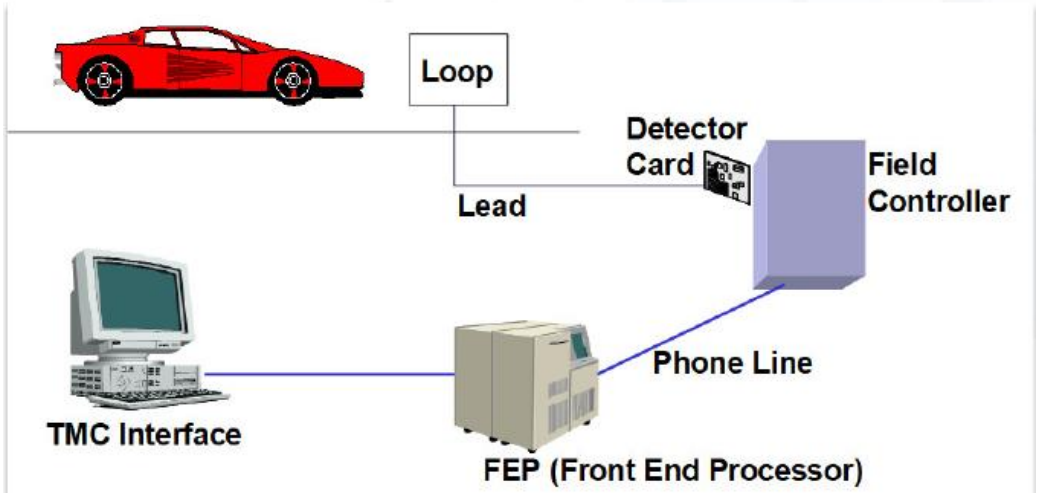
image borrowed from Richard Johansson (*Chalmers Technical University and University of Gothenburg)*

Introduction to Traffic Flow Data

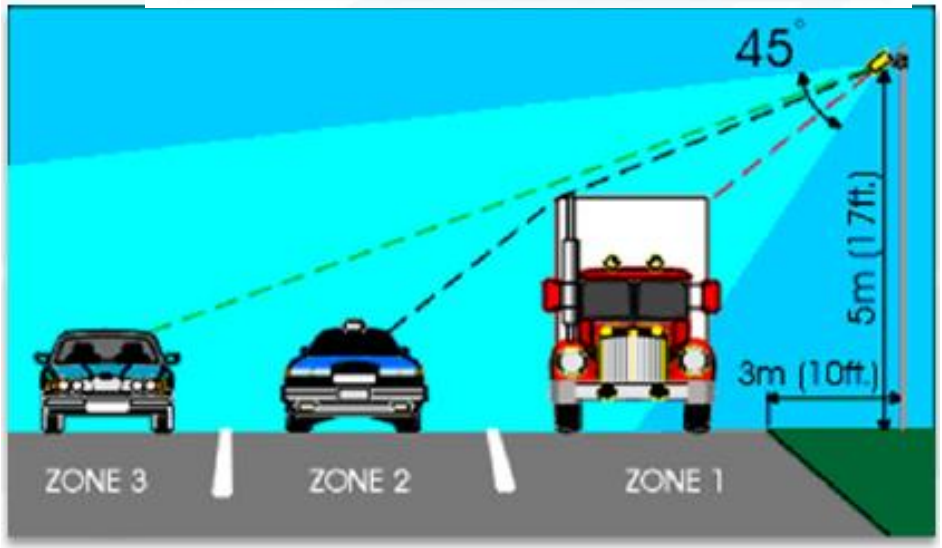
Traffic Sensor : Loop Detection

루프 검지기

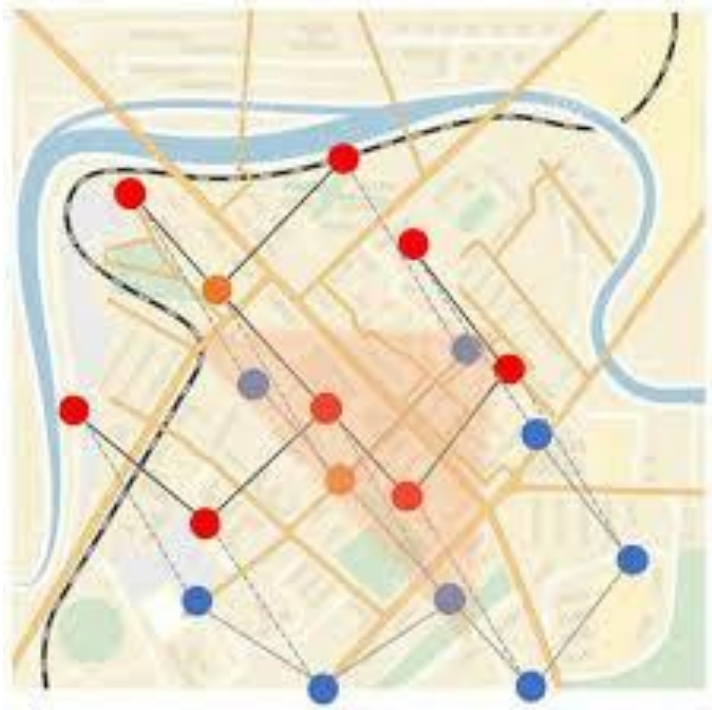
- 교통량 산정
- 도로 용량 분석
- 차량 통과 속도 감시

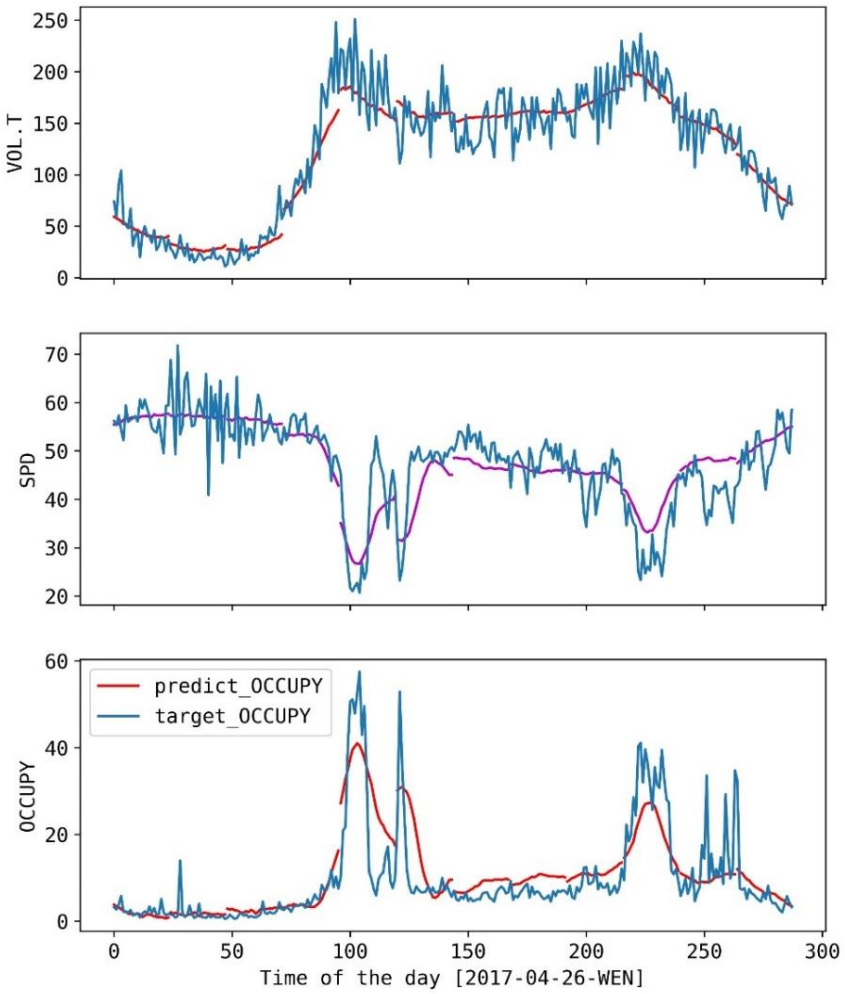


교통 레이다



VDS 데이터 전처리 및 활용 : RNN 예측



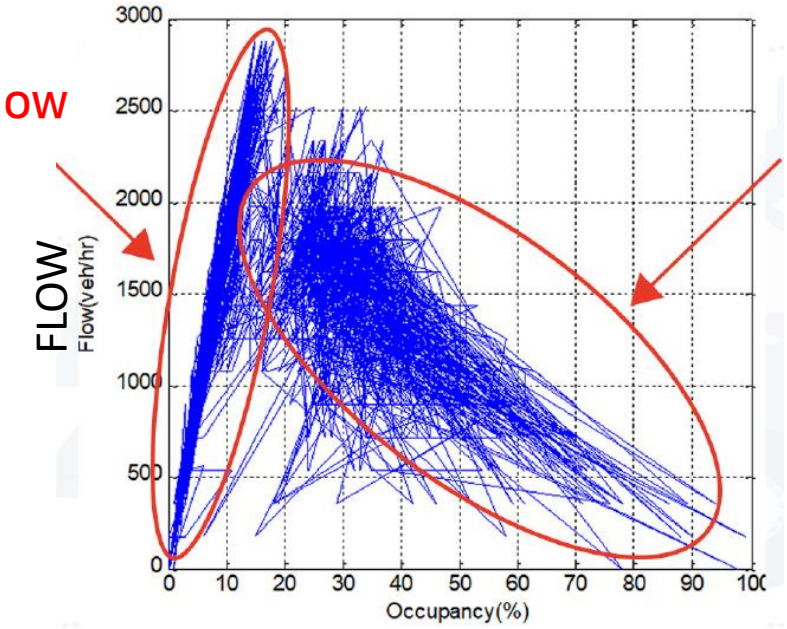


Traffic Flow Theory

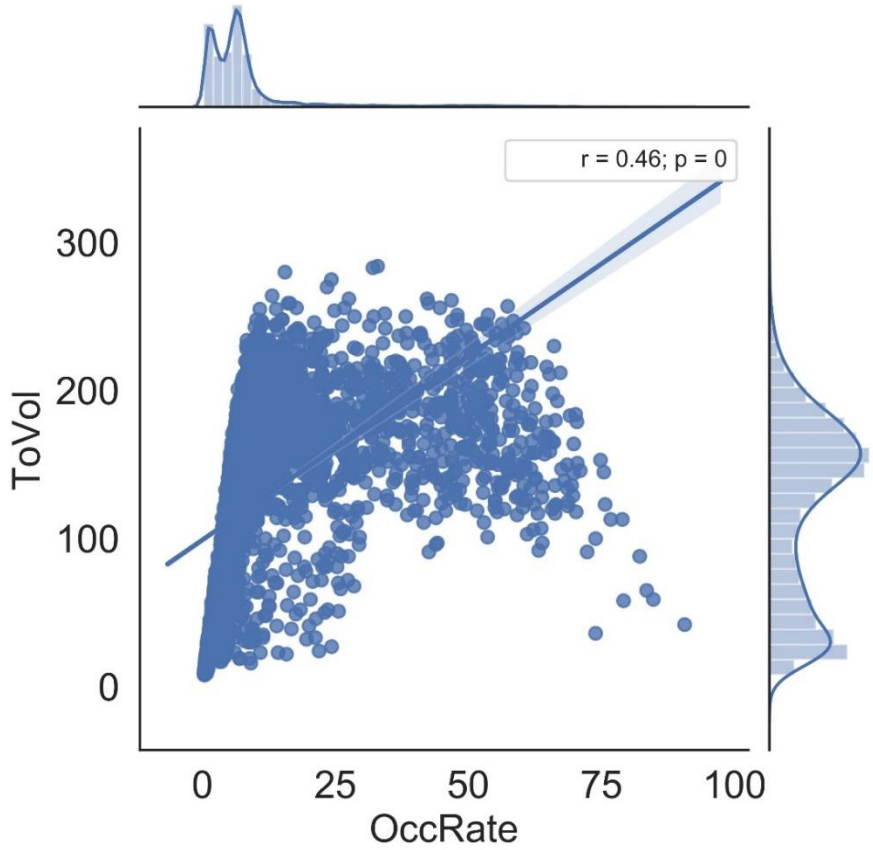
Daejeon City VDS Data (#17)

Traffic Jam

Free Flow

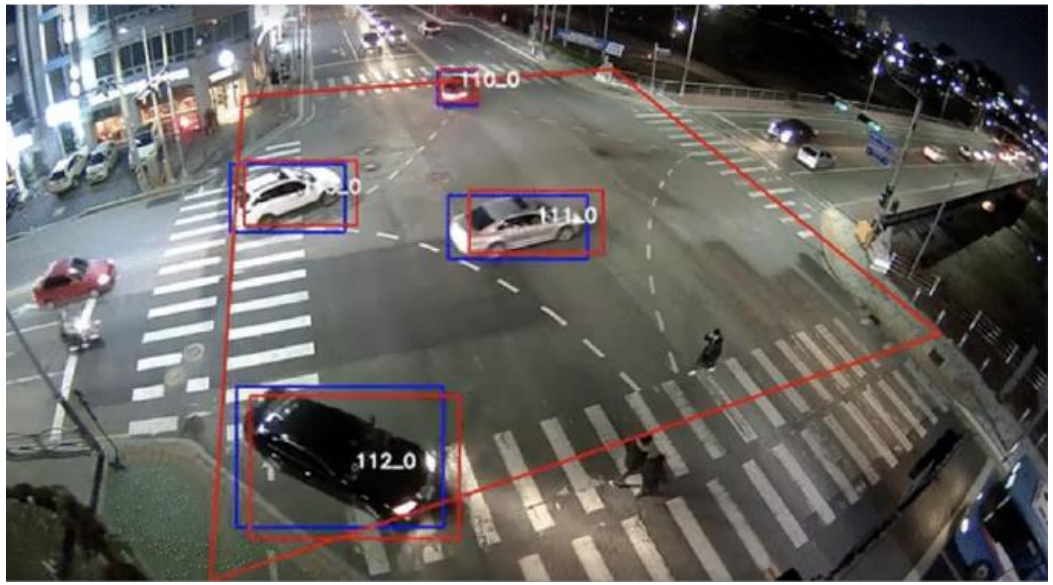


OCCUPANCY



CCTV : Object Detection

영상 검지기

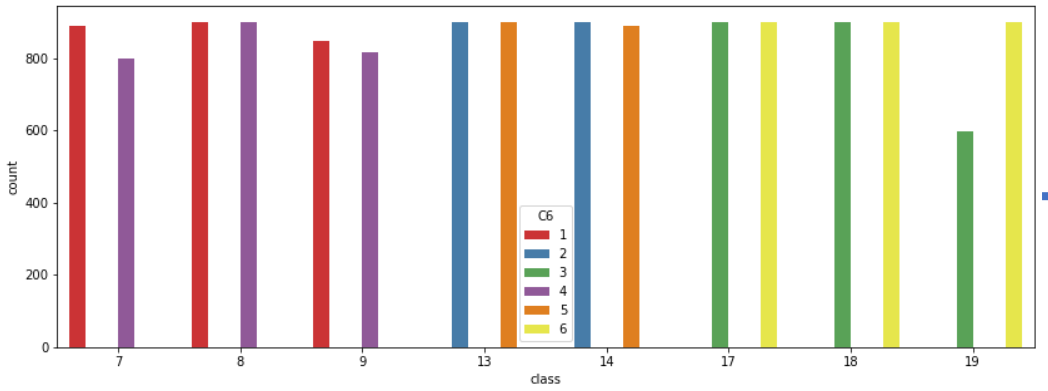


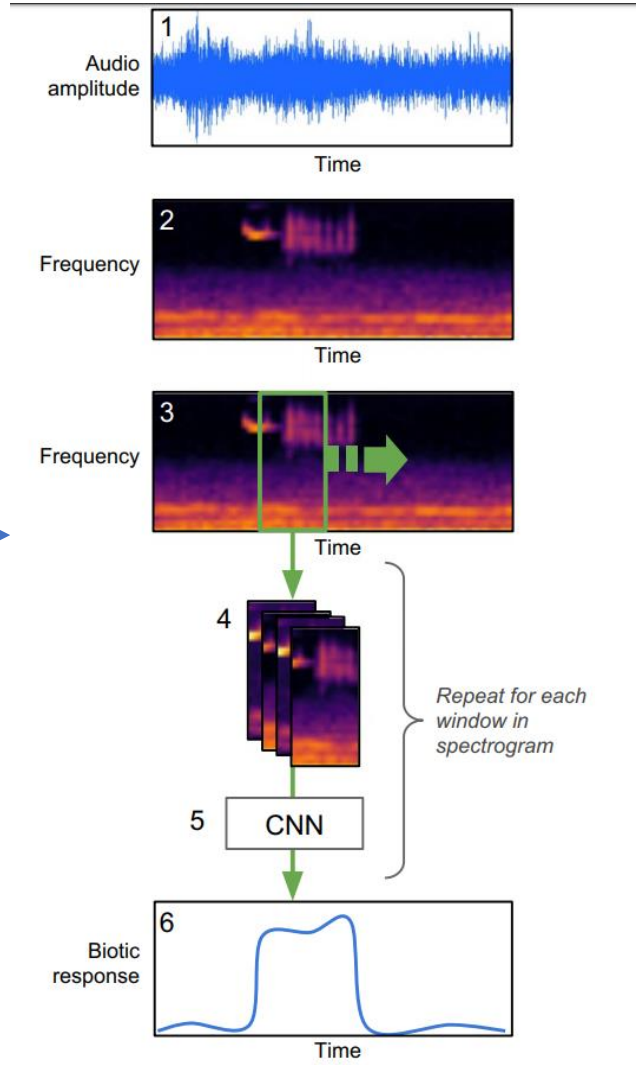
RSU(Road Side Unit)

도로를 운행하는 차량에 설치된 단말기와 WAVE 무선통신을 수행 차량 단말기에서 전송하는 각종 정보를 수집 저장하여 센터로 전송하는 기능

DJ_TrafficSound14K and CNN Classification

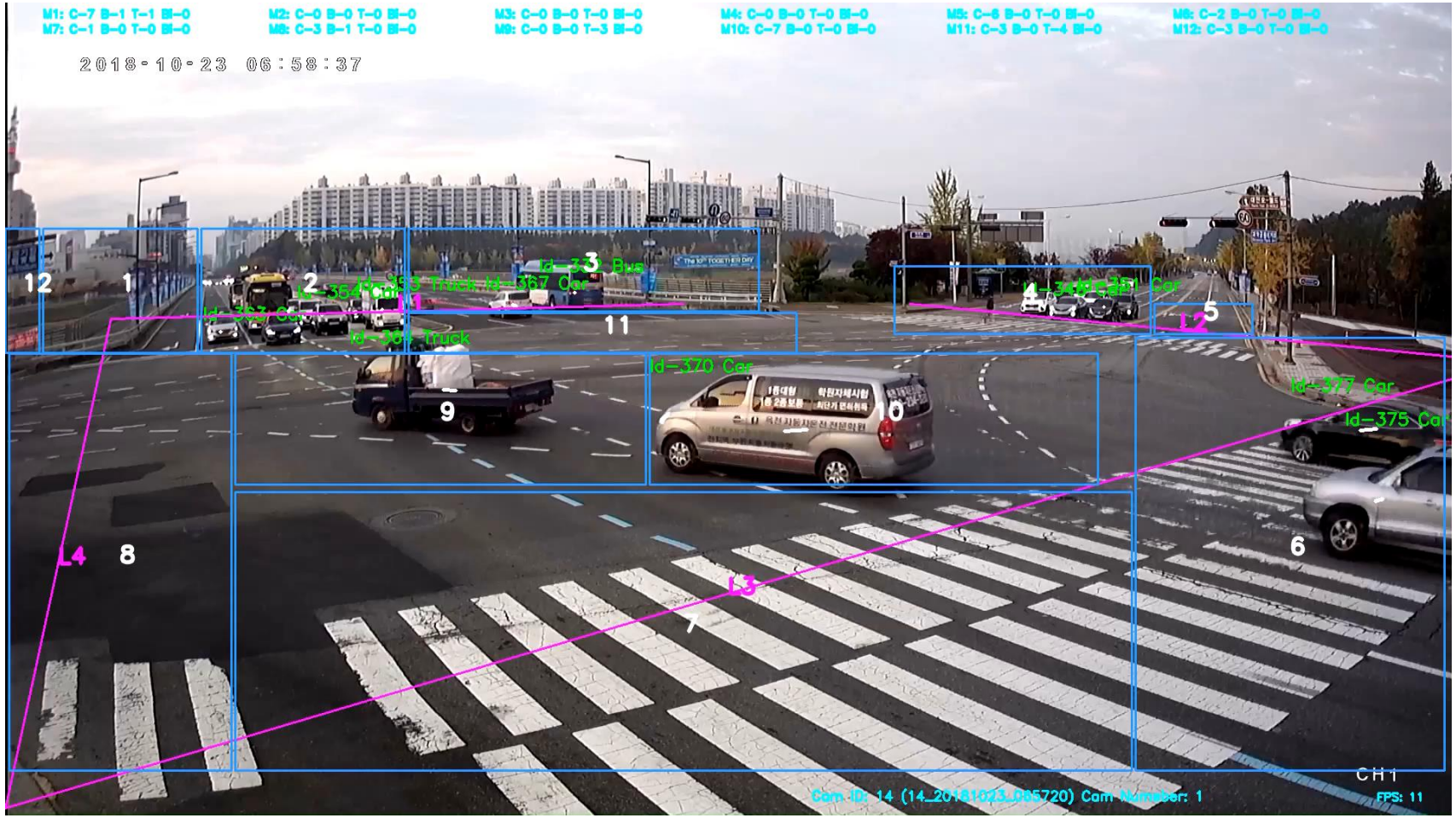
- 소음 빅데이터 자체 수집 : DJ_TrafficSound14K (대덕대로)





YOLO : Vehicle Detection and Counting

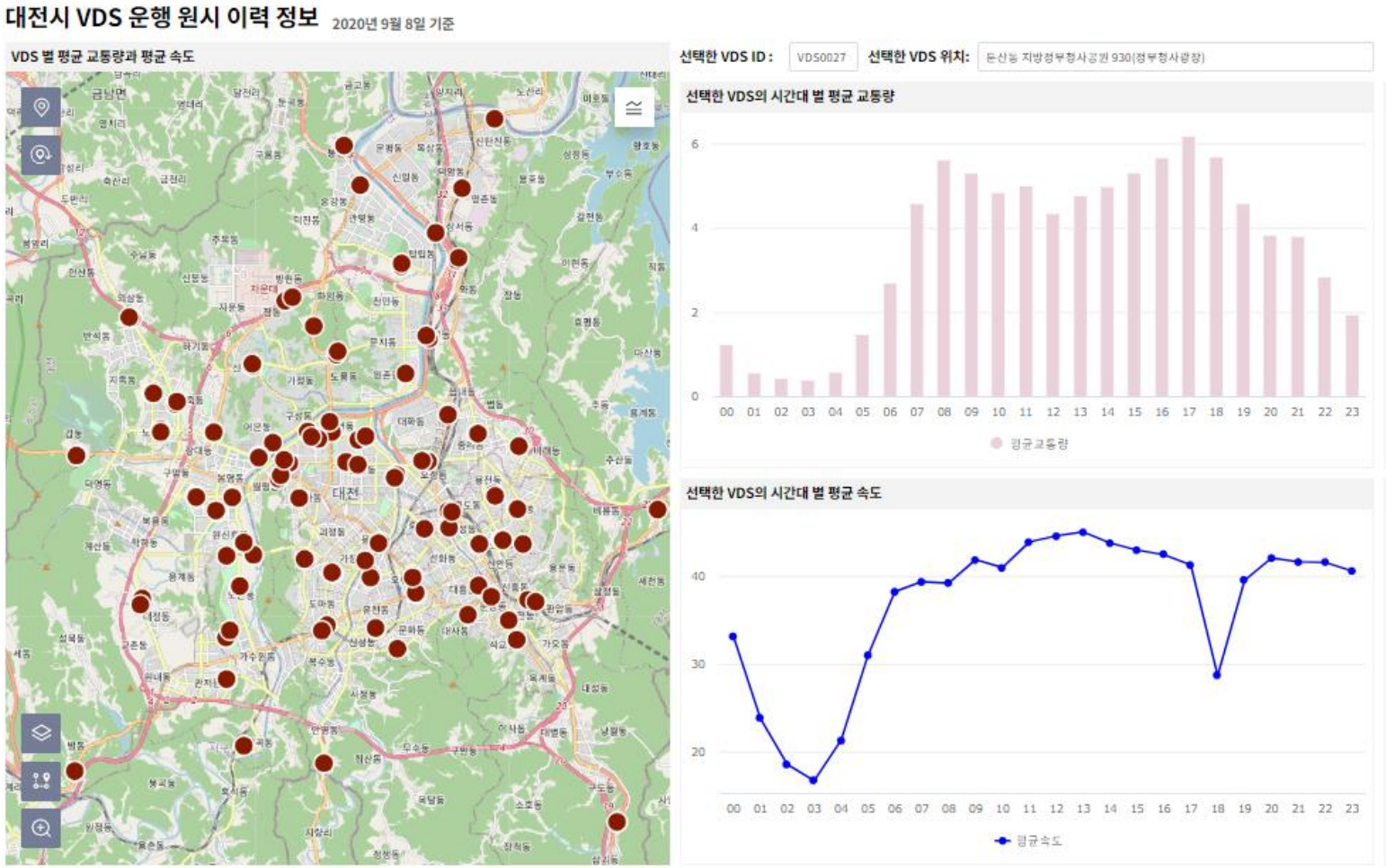
- CVPR20의 'AI City Challenge' Competion (7th, KISTI)



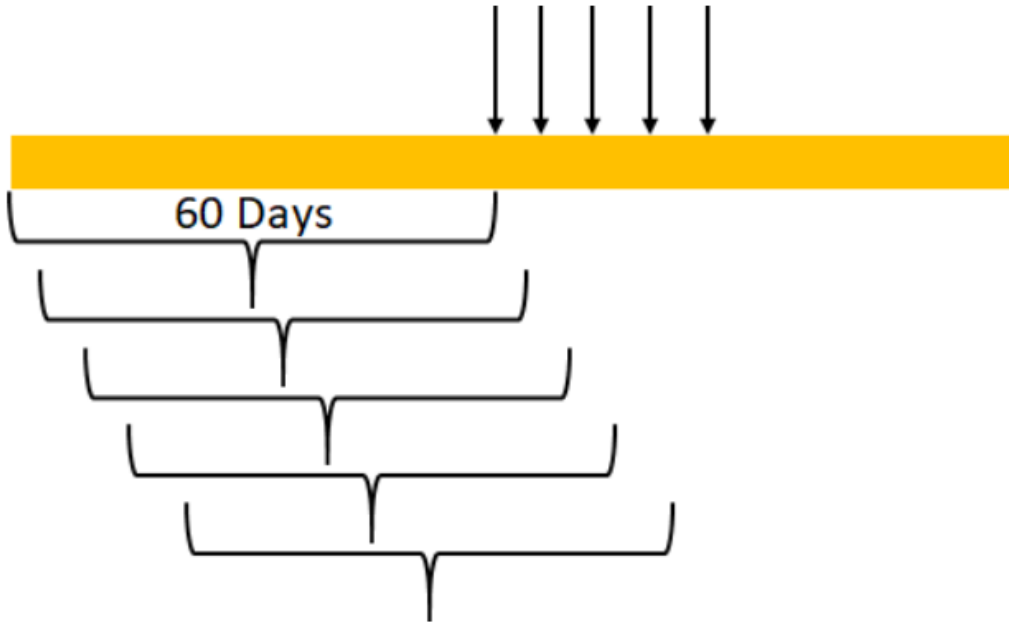
교통 데이터 다운로드

❖ 국가교통데이터 오픈마켓

https://www.bigdata-transportation.kr/



LSTM을 이용한 교통 흐름 예측

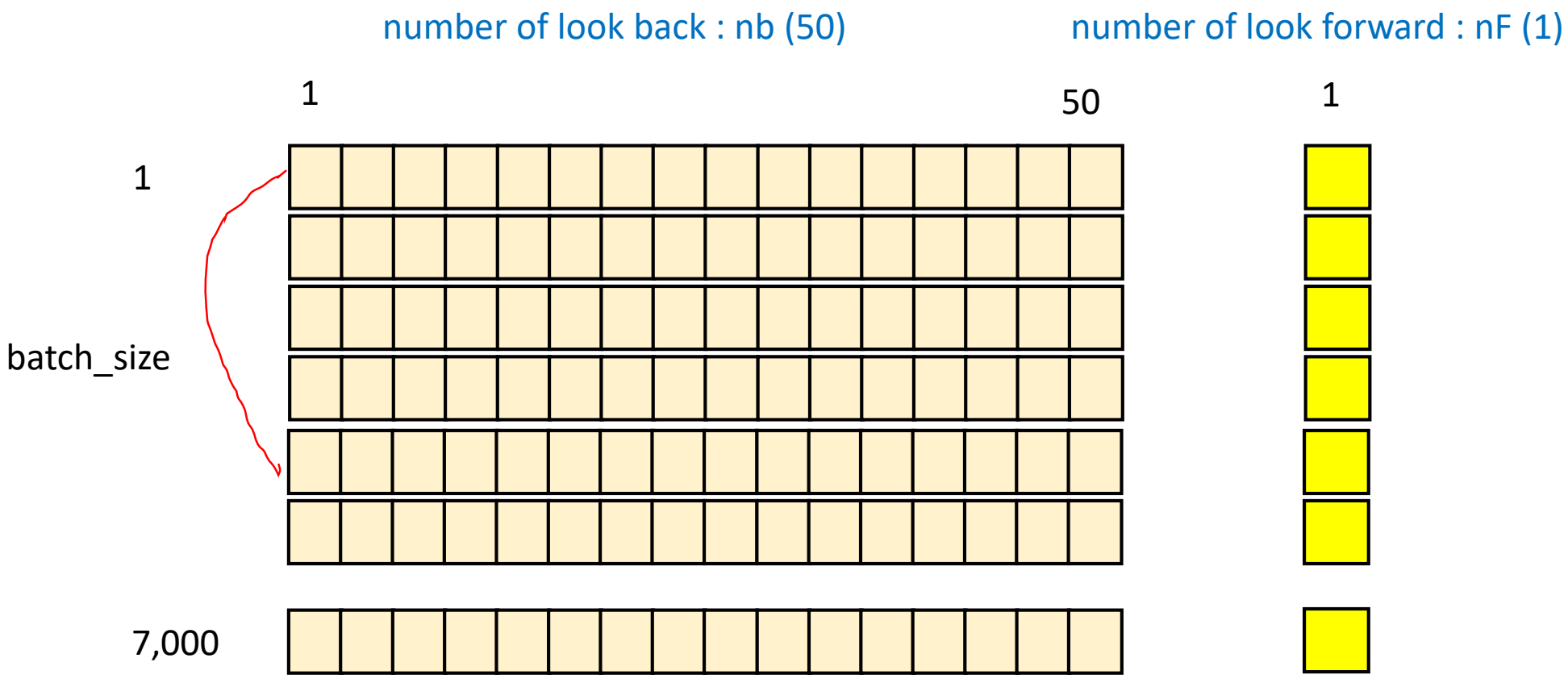


VDS 데이터를 이용한 교통 흐름 예측 모델 단계

- Data Preprocessing
- Building the RNN
- Making the prediction and visualization

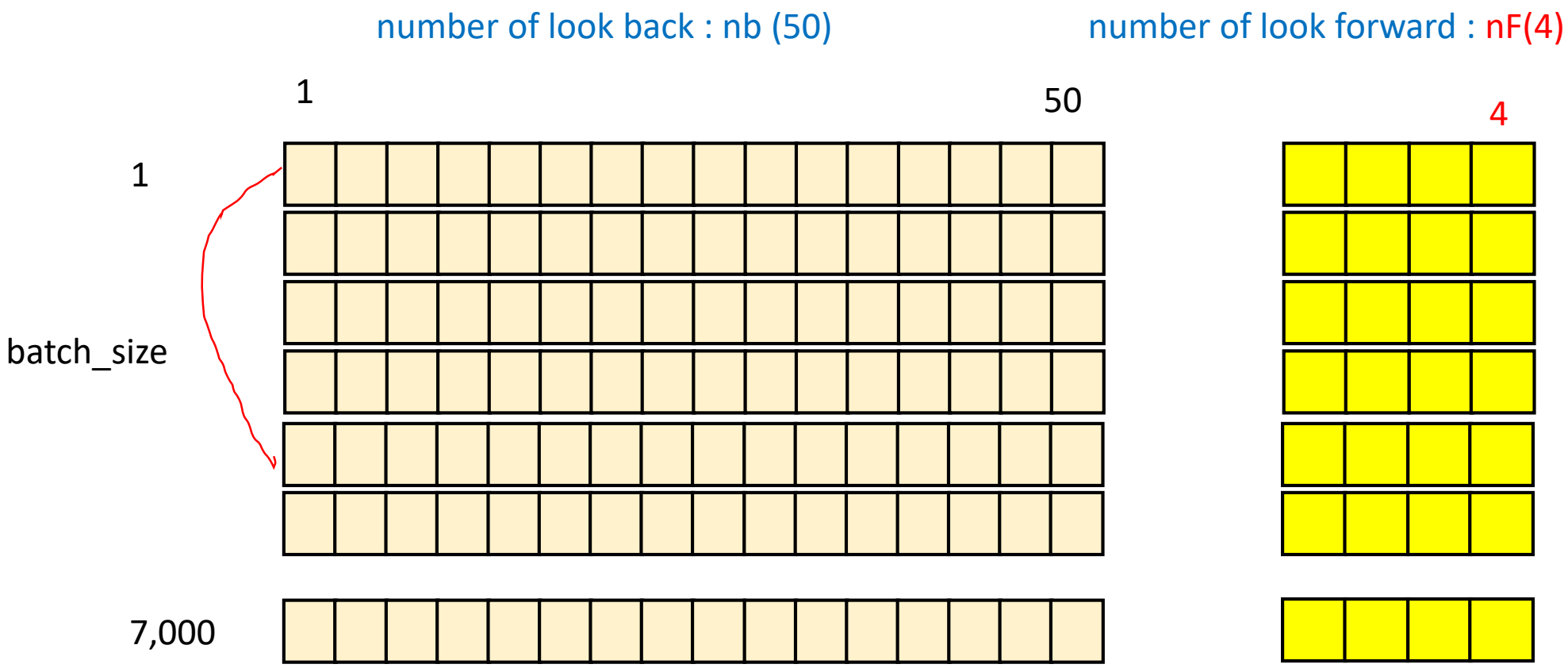
Many-to-One RNN Data Structure

X_train[7000,50,1] y_train[7000,1,1]

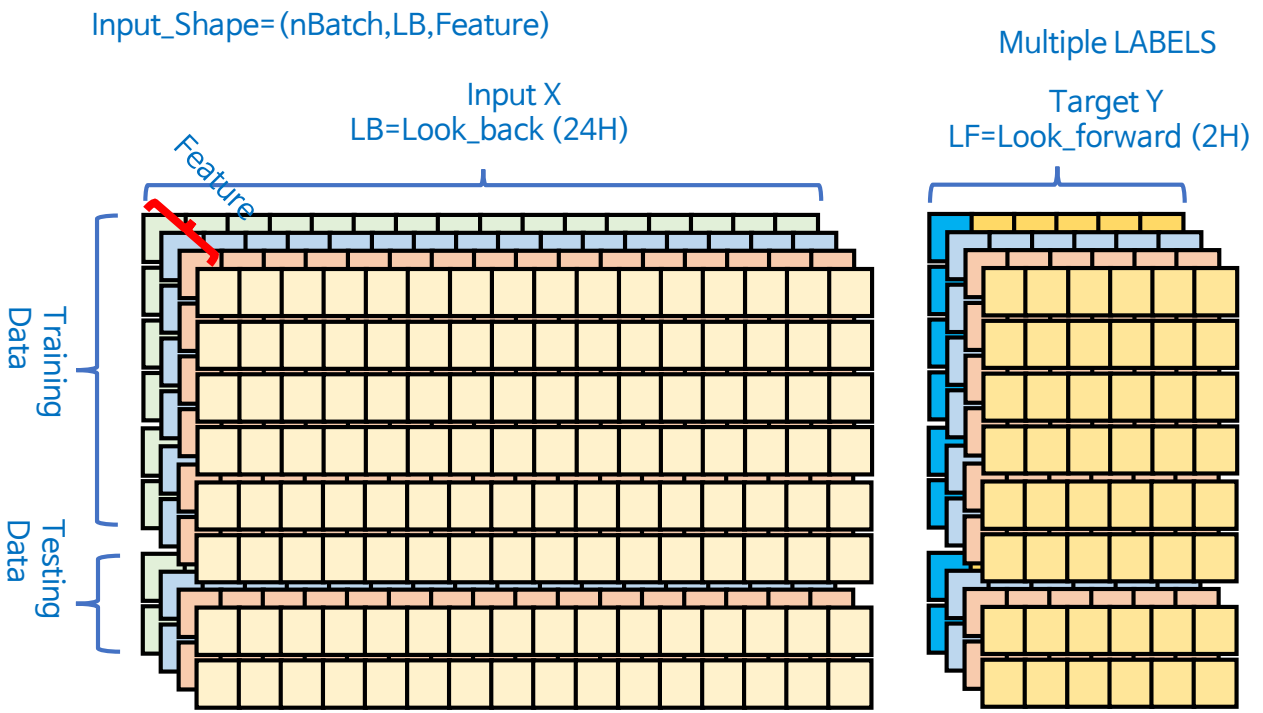


Many-to-Many RNN Data Structure

X_train[7000,50,1] y_train[7000,nF,1]



RNN Input-Output Data Structure

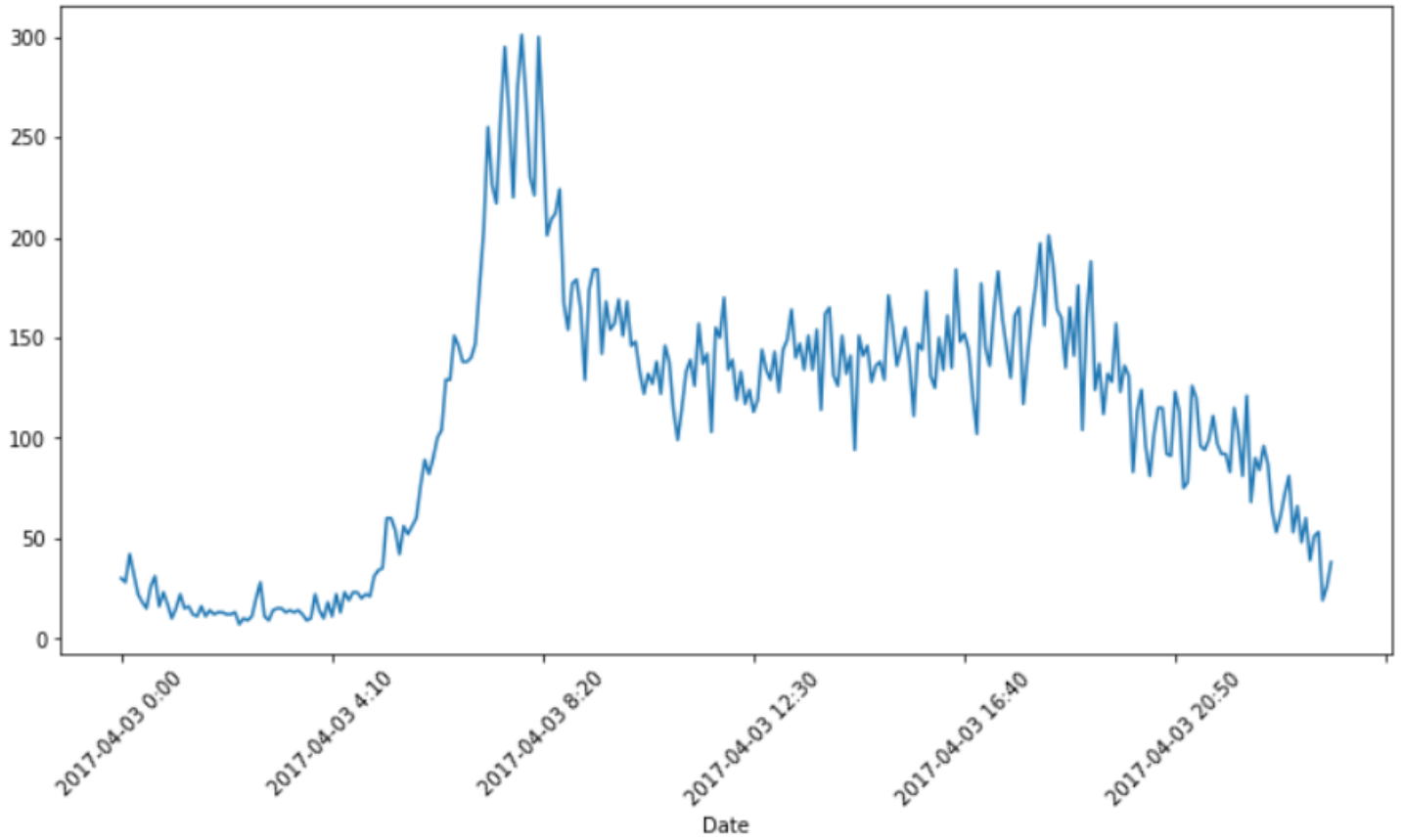


VDS, RSE Data Preprocessing for LSTM

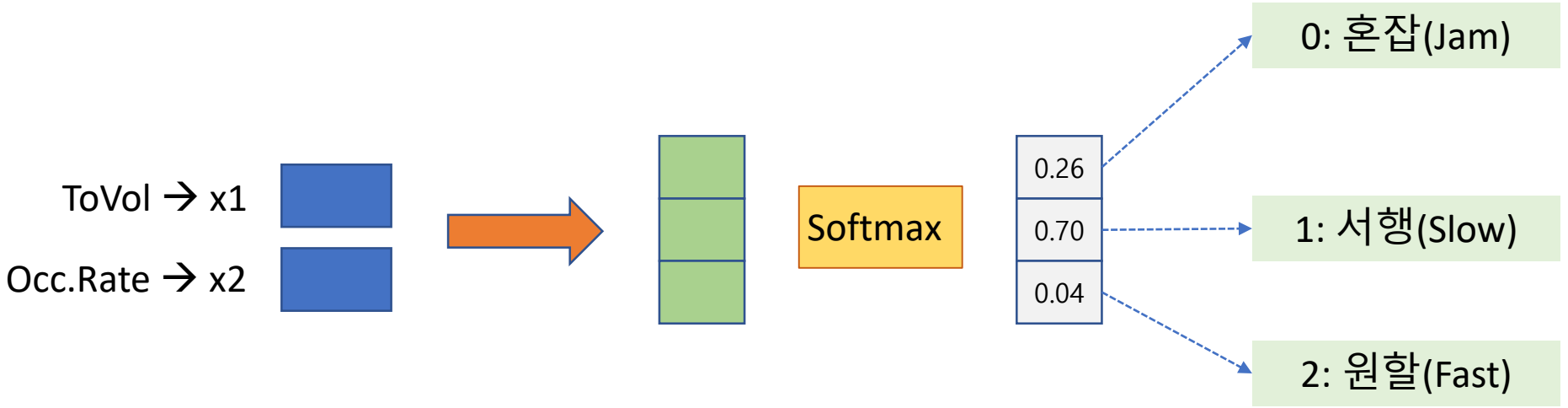
판다스로 데이터 전처리

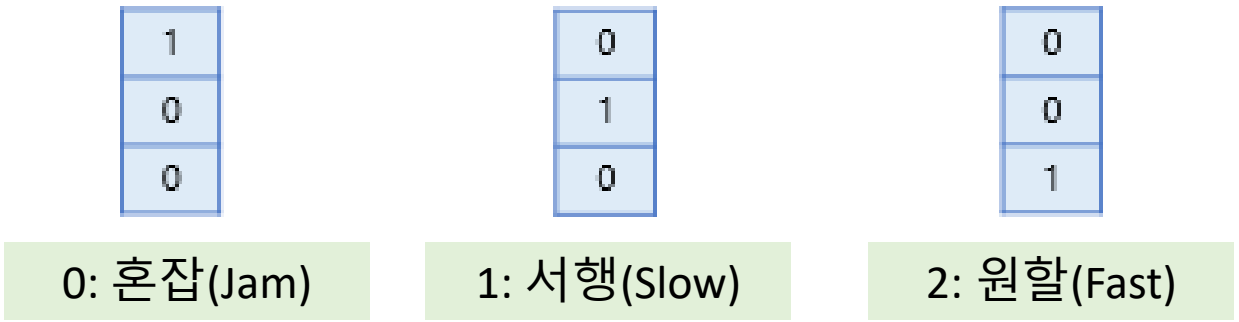
In [17]:
```
## 2017년 4월 3일은 월요일.
df['ToVol'][288:576].plot(rot=45,figsize=(12,6))
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x299414bb9c8>



라벨이 3개 일경우


ToVol → x1
Occ.Rate → x2
Softmax
0.26
0.70
0.04
0: 혼잡(Jam)
1: 서행(Slow)
2: 원할(Fast)


1
0
0
0: 혼잡(Jam)
0
1
0
1: 서행(Slow)
0
0
1
2: 원할(Fast)

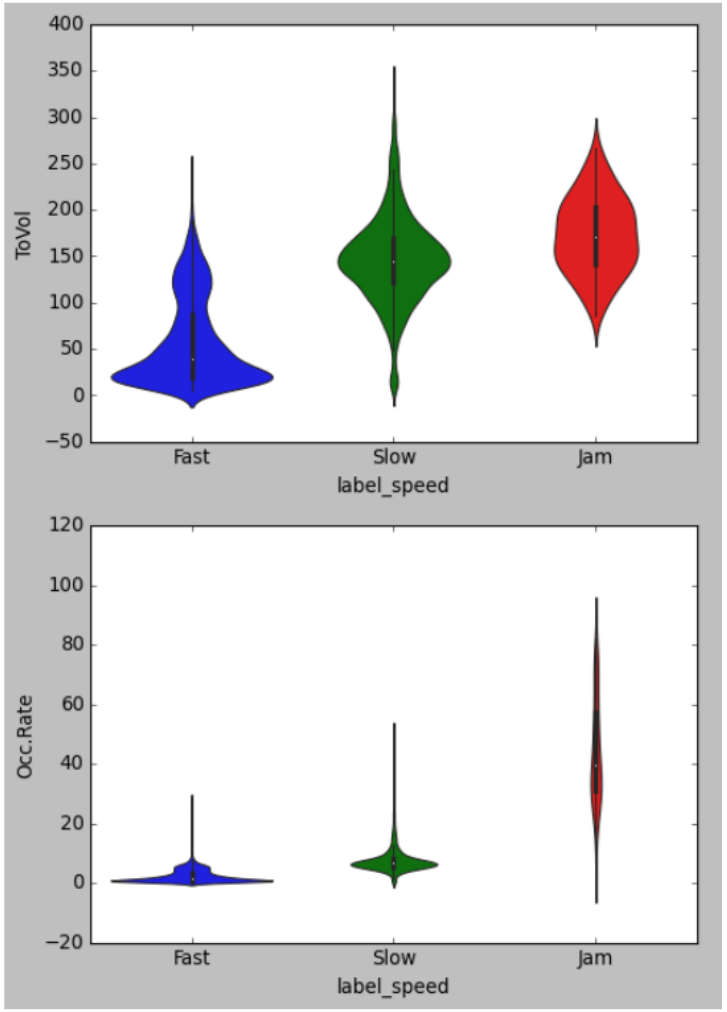
데이터 가시화

In [17]:
```
def get_score(v):
    if v < 30:
        score = 'Jam'
    elif v < 50:
        score = 'Slow'
    else :
        score = 'Fast'
    return score
```

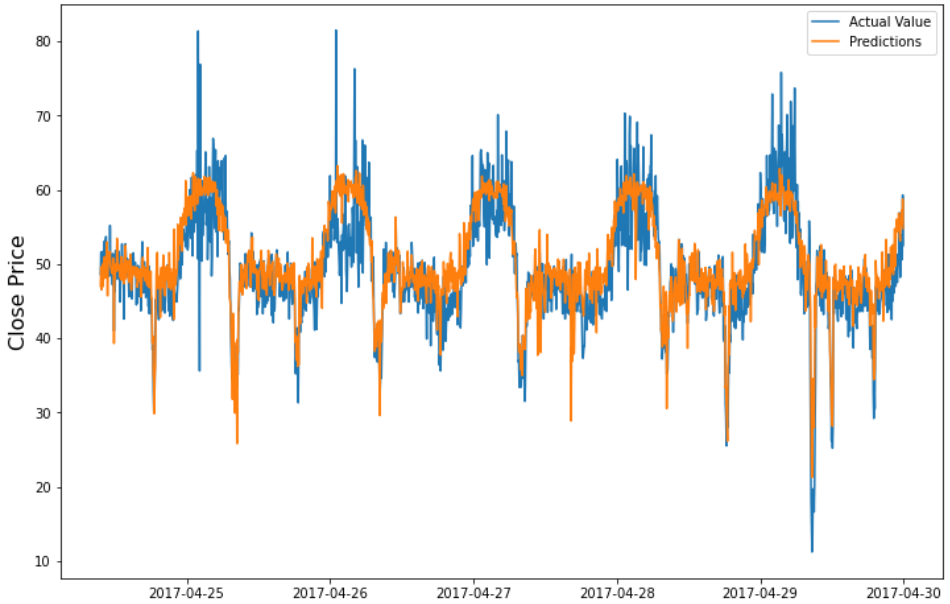
In [18]:
```
df["label_speed"] = df["Speed"].apply(lambda v: get_score(v))
df.head(4)
```

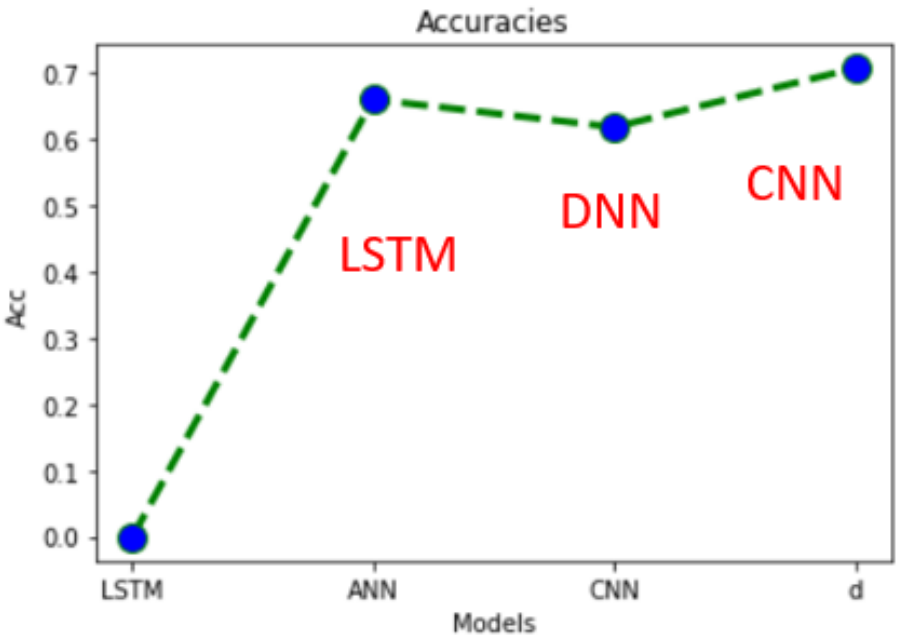
Out[18]:

Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	label_speed
2017-04-02 0:00	43	34	9	0	50.3	1.90	Fast
2017-04-02 0:05	45	32	13	0	58.9	1.84	Fast
2017-04-02 0:10	46	34	12	0	50.6	1.87	Fast
2017-04-02 0:15	45	36	9	0	50.9	1.72	Fast



A3: 시계열 패턴을 무시한 1D CNN 회귀





Transformer Model for Traffic Flow Prediction

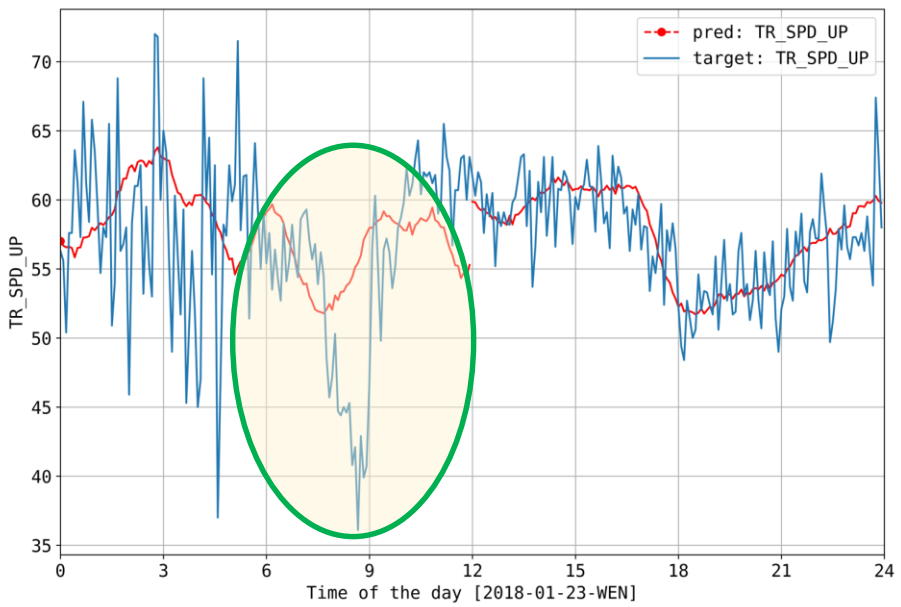
대상 도로 : 대학로 (유성구청네거리-구성삼거리)

❖ 약 1.4km 왕복 4차로, RSE 3개, 영상검지기 5개(현재 3개, 향후 2개)

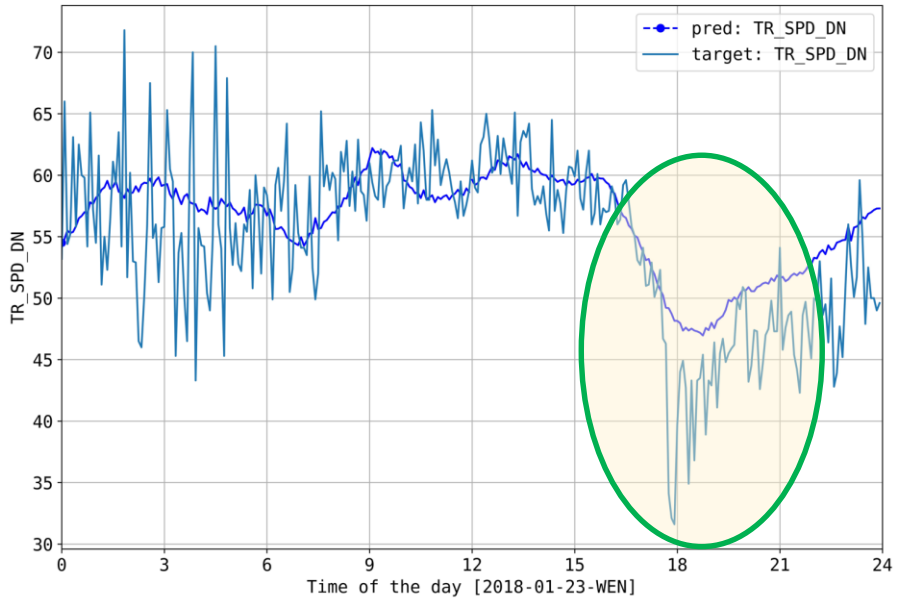
LSTM을 이용한 교통 흐름 예측 및 새로운 모델의 필요

LSTM 예측은 RNN에 비하여 성능은 좋은나, Long-Sequence 장기예측에는 성능저하가 발생한다.

상행 12시간예측 (2018.01.23.,수요일)



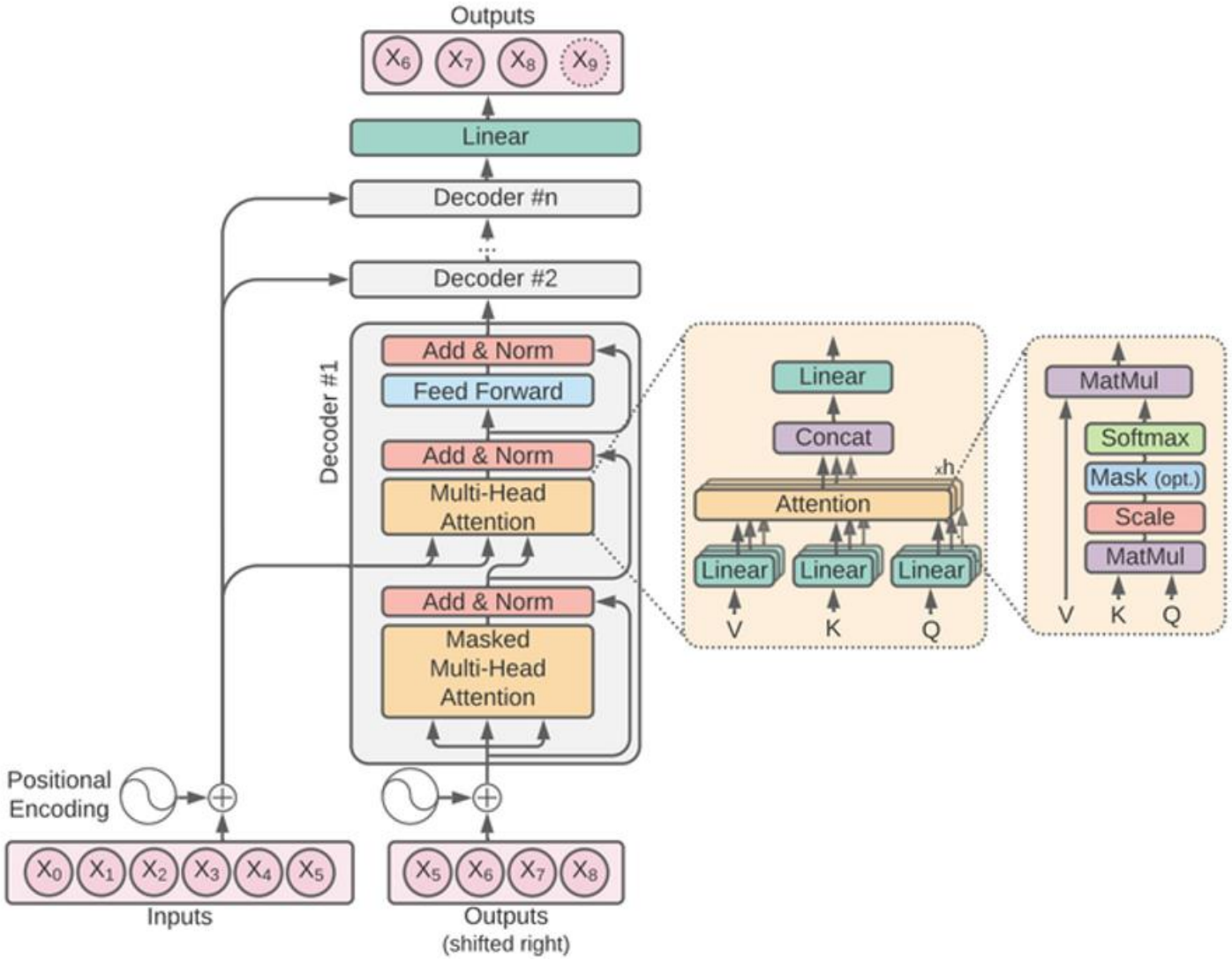
하행 12시간예측 (2018.01.23.,수요일)



LSTM 장기예측(12시간)과 예측 정확은 재고할 필요가 있다. → 새로운 모델이 필요하다.

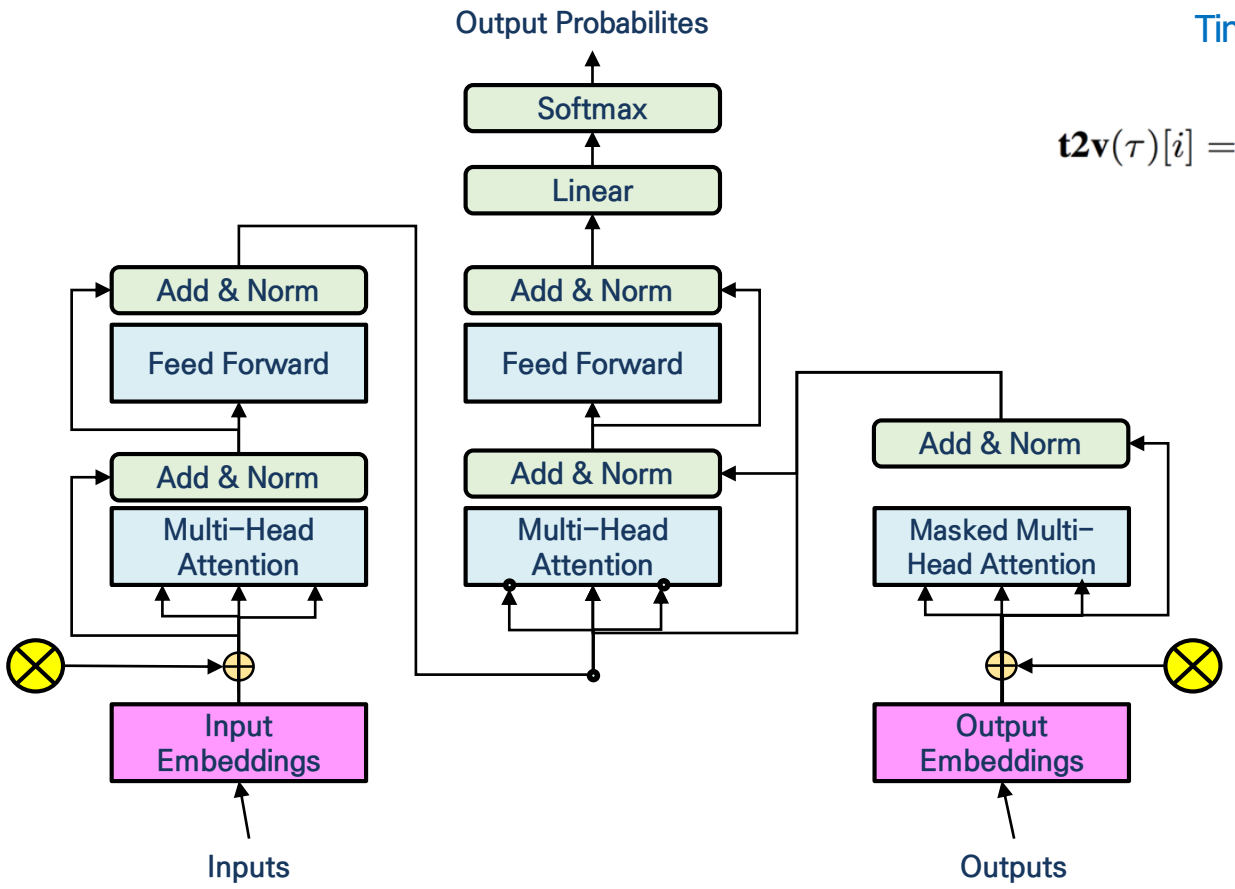
☆ LSTM 장기 예측(12시간) 결과는 상대적으로 정확도가 현저히 떨어진다.

Multi-head Attention Mechaism



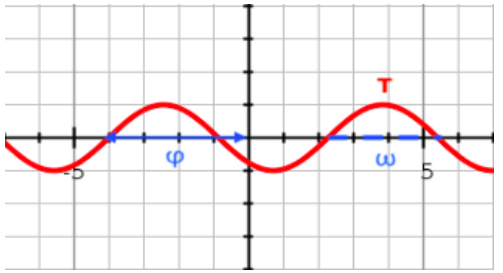
Position Encoding : Time2Vec

자연어 처리 트랜스포머 모델의 포지셔널 인코딩을 교통에 적합하게 Time2Vec 개발이 핵심이다.



❖ Attention 도입으로 시계열성을 없어지고, 대신에 Time2Vec 임베딩을 이용하여 시공간적인 정보 유지

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$



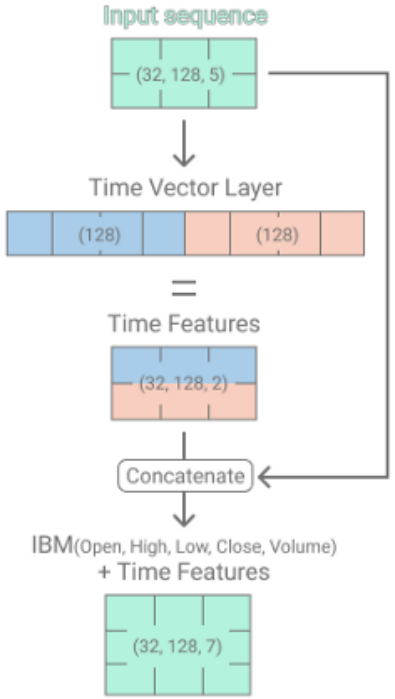
주기성: 삼각함수(sin)를 사용

비주기성: 선형함수

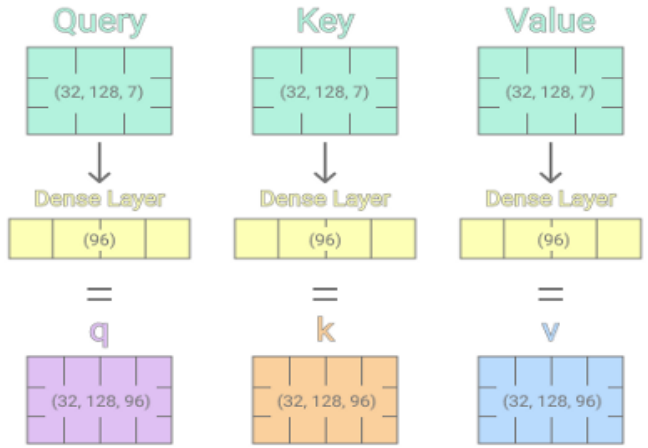
Attention : Query, Key, Value

Time2Vec

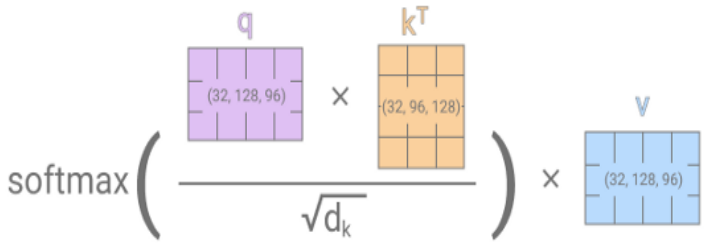
- 5개 VDS 17, VDS 18, VDS 19, VDS 20, VDS 21
- 트랜스포머 모델 입력크기(input-size)
  - batch_size: 32, sequence_length: 128, feature : 5



Single Head Attention : Query, Key, Value



- Attention Weight는 Softmax를 사용함. Q*K*V

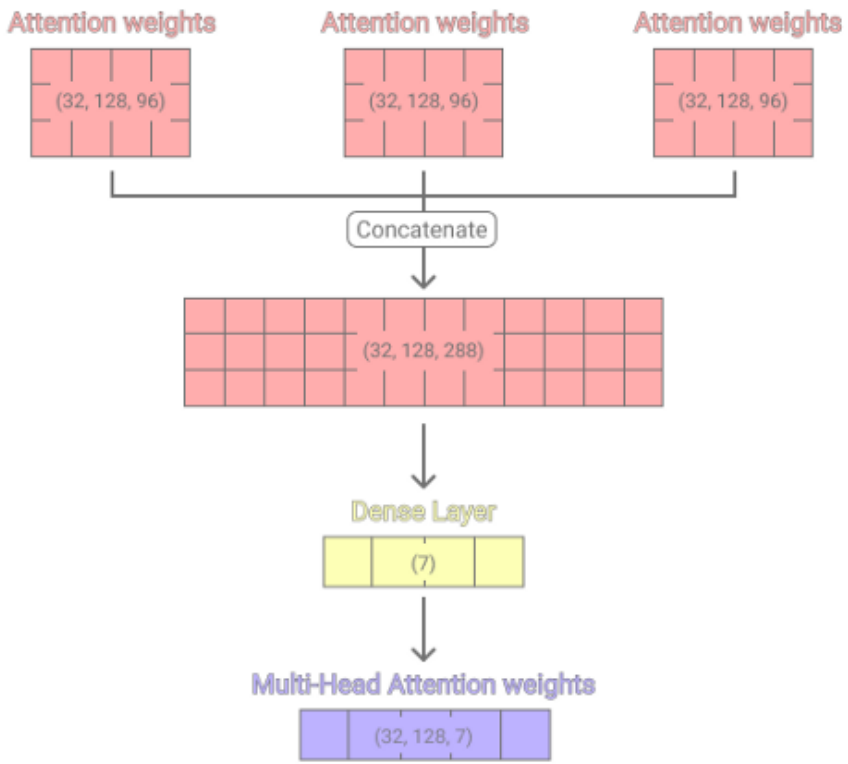


Transformer Traffic Model

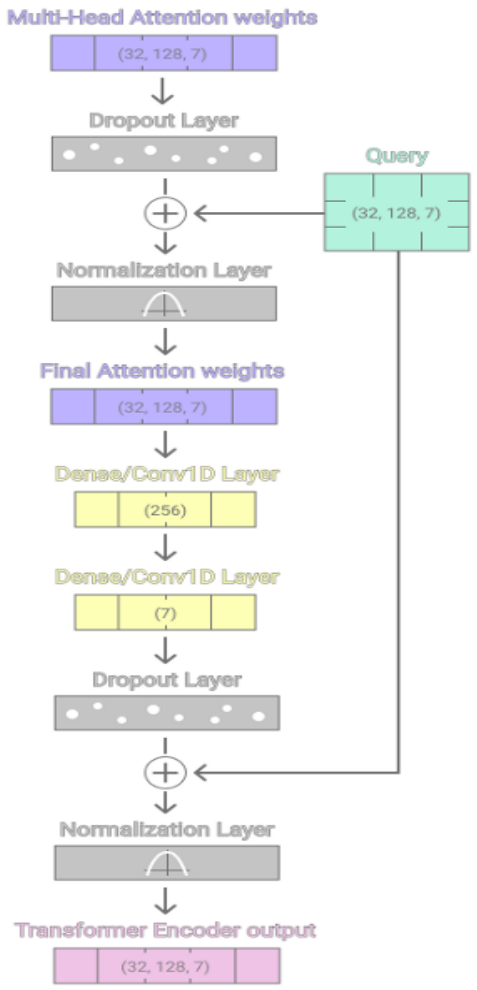
KISTI SINCE 1962

Mulit-Head Attention으로 병렬처리가 가능함

❖ VDS 17, VDS 18 데이터 구조 (train, test, validation)

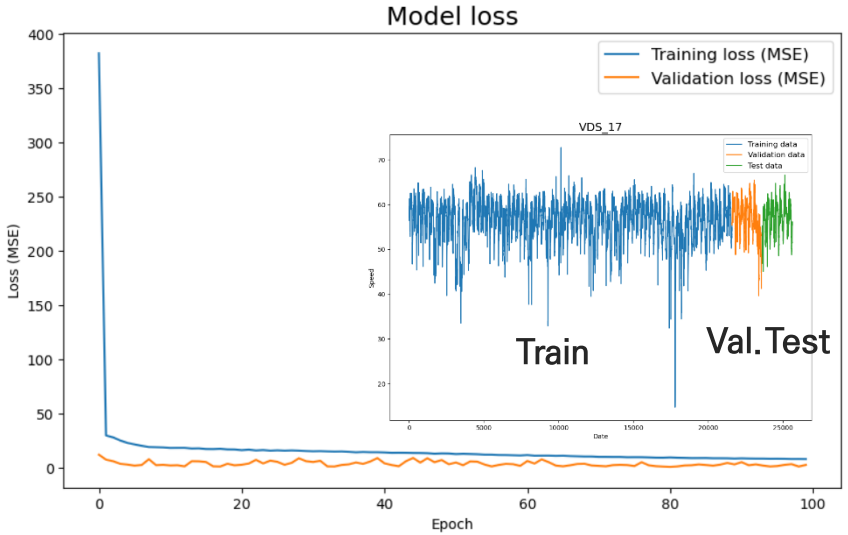


Transformer Encoder 출력이되 Decoder에 사용됨



Long-Term prediction of Traffic Speed and Volume

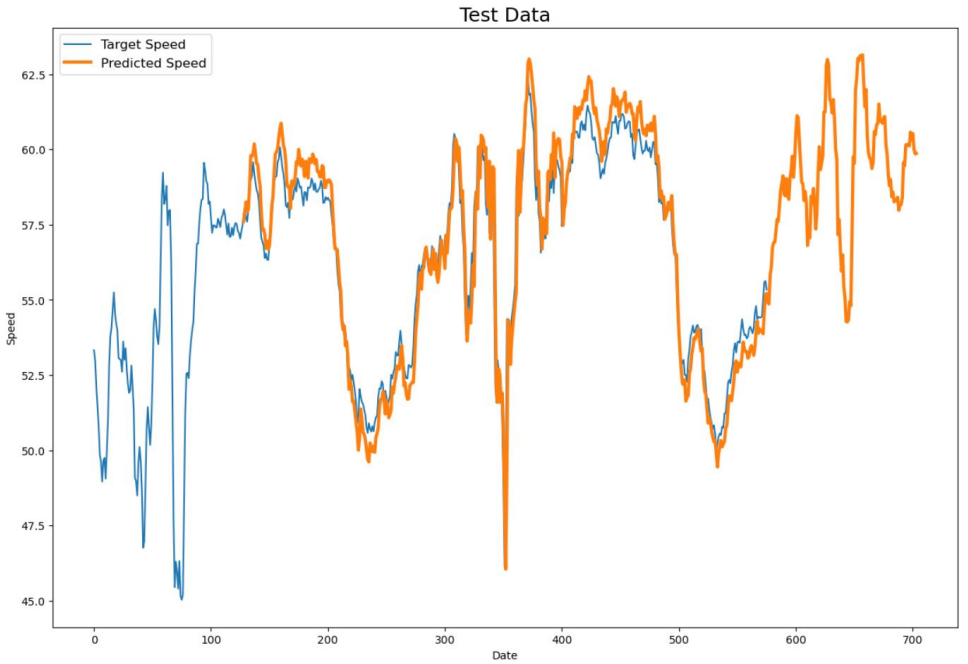
대전시 차량검지기(VDS) 데이터 105개



- 대학로 관련 5개 위치 데이터 90일
  - 데이터 개수: 25,920개
  - 90일*24*12 (5분 단위)

장기예측 성능 : 2016개 예측(1주일, 288*7)

- TransformerVDS: Transformer+TimeEmbedding



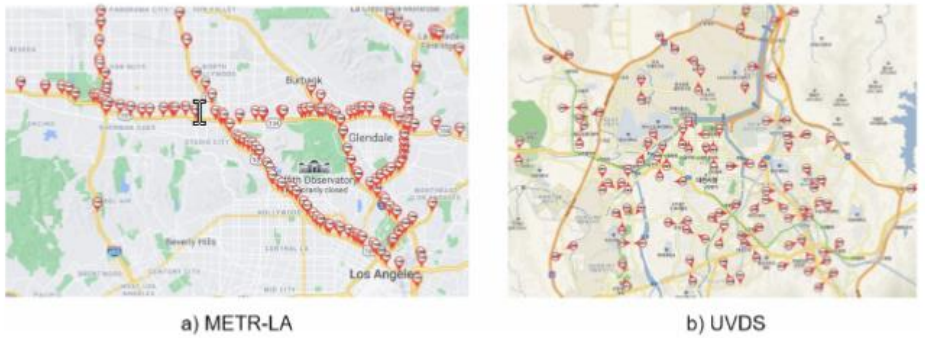
- TransformerVDS 결과와 대전 UVDS 데이터에 적용하고 있으며, SCI 논문에 투고할 예정

Dynamic Spatial Transformer WaveNet Network

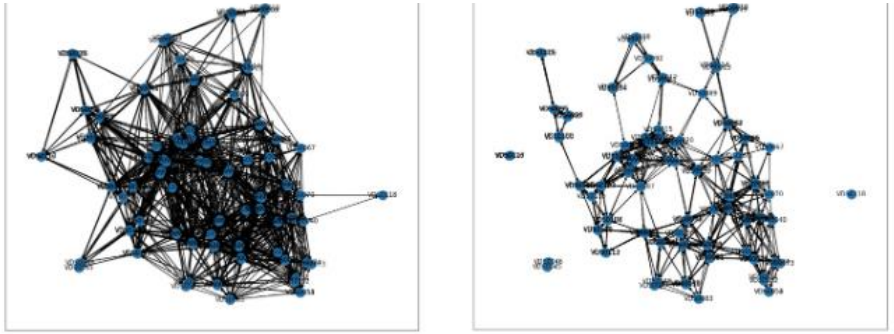
KISTI SINCE 1962

UVDS: 대전 VDS 데이터 기반 시공간 그래프 신경망 구축
Spatial-temporal Graph neural Network

- Metr-LA : 대표적 교통 공개 데이터 (미국, LA지역 4개월)
- UVDS 공개 : (시공간) 대전시 차량검지기 데이터 (3개월)



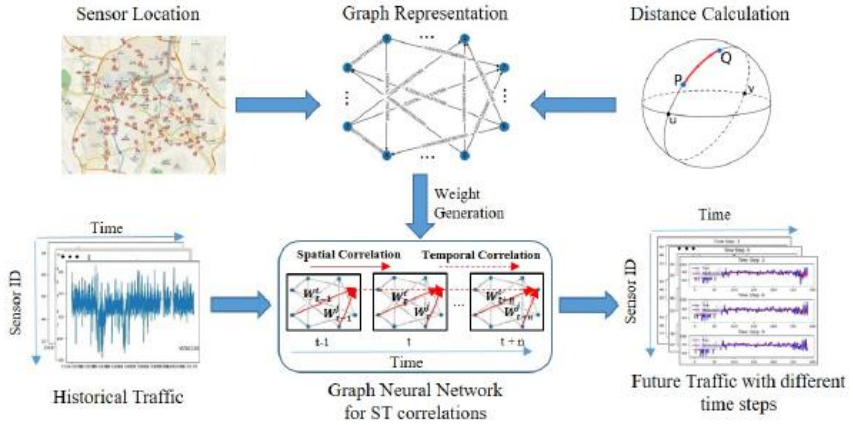
a) METR-LA b) UVDS



a) β = 0.1 b) β = 0.5

대전시 VDS 데이터 기반 새로운 UVDS 데이터 공개
UVDS: A New Dataset for Trac Forecasting with Spatial-Temporal Correlation

- General Framework for Traffic Forecasting using Spatial-temporal Correlaitons



Dynamic Spatial Transformer WaveNet Network

UVDS 데이터 기반 동적 시공간-트랜스포머 웨이브넷 신경망 개발 및 UVDS 데이터 성능 테스트
DSTWN 모델 : Dynamic Spatial Transformer WaveNet Network로 Spatial-Temporal Graph 신경망 보다 우수성 검증

❖ 기존 Spatial-Temporal Graph Network과 비교 성능 향상 목표

❖ 개발한 DSTWN 알고리즘 성능 벤치마크 : Metr-LA 데이터

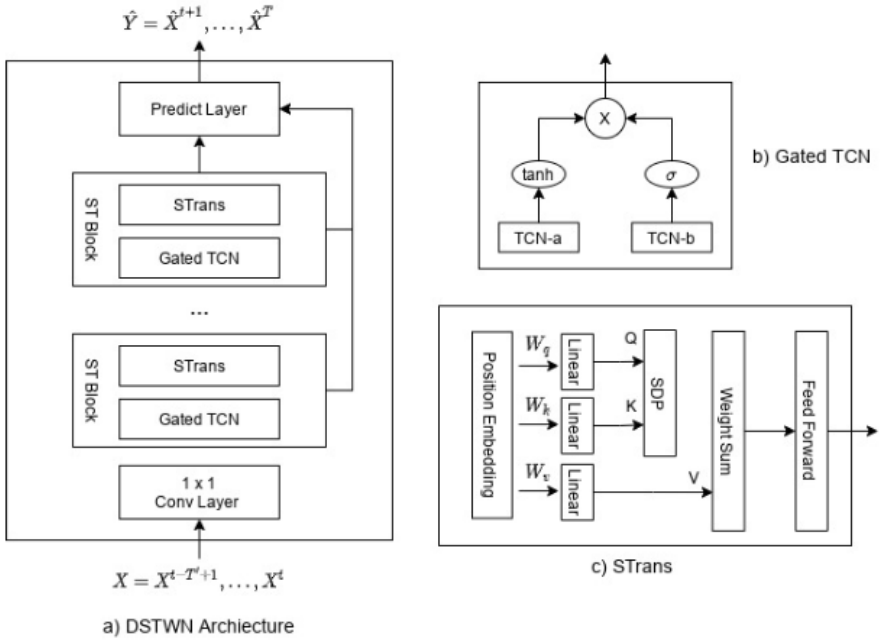


Fig. 3. The DSTWN Architecture

Table 2. Results on METR-LA dataset.

Step	Metrics	GraphWN	MTGNN	STWaNet	STTN	DSTWN
1	MAE	2.2408	2.2441	2.2791	2.4080	2.3220
	RMSE	3.8636	3.9185	3.9793	4.2339	4.0704
	MAPE	0.0540	0.0551	0.0550	0.0588	0.0567
3	MAE	2.7127	2.6762	2.7457	2.9160	2.8758
	RMSE	5.1690	5.1428	5.3162	5.6556	5.6285
	MAPE	0.0695	0.0686	0.0711	0.0778	0.0759
6	MAE	3.0974	3.0605	3.0947	3.3819	3.2909
	RMSE	6.1839	6.2002	6.2781	6.8651	6.7740
	MAPE	0.0847	0.0816	0.0838	0.0958	0.0900
9	MAE	3.3617	3.3100	3.3239	3.6961	3.5461
	RMSE	6.8279	6.8380	6.8639	7.5749	7.3837
	MAPE	0.0950	0.0912	0.0924	0.1085	0.0984
12	MAE	3.5760	3.4937	3.5036	3.9533	3.7446
	RMSE	7.2883	7.2421	7.2761	8.1262	7.8246
	MAPE	0.1035	0.0982	0.0993	0.1181	0.1047
Training		45.6927	62.8770	54.5744	77.5496	133.2374
Inference		1.4630	1.6825	1.5830	6.6945	3.8088

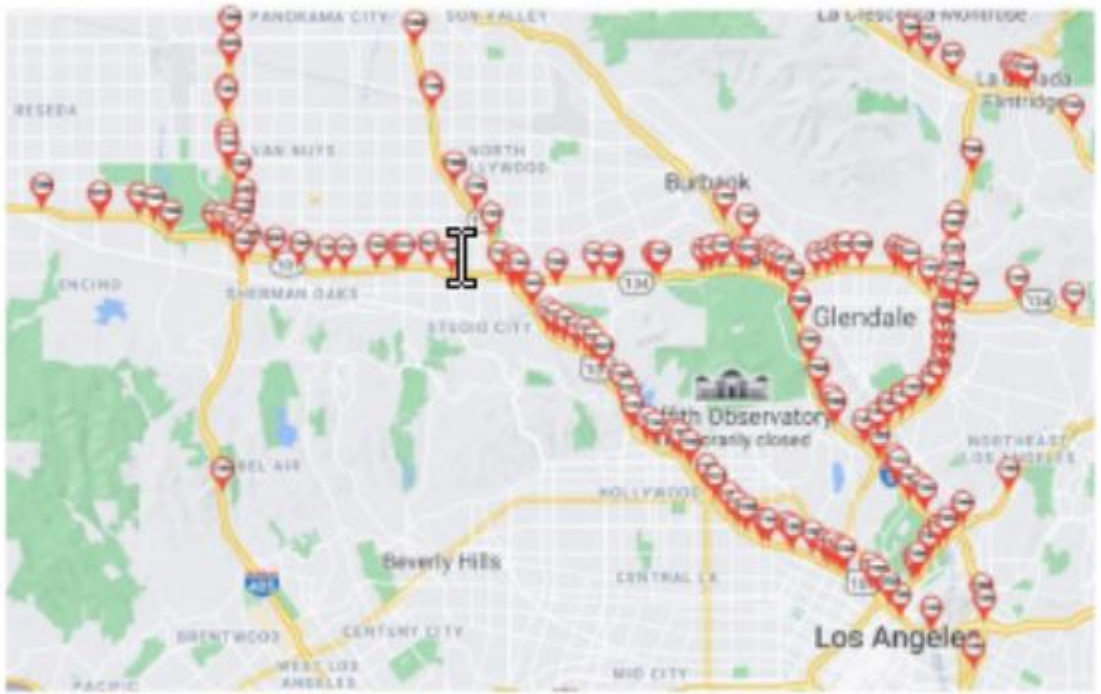
Public spatial–temporal tra±c datasets

Table 1. Public datasets for spatial–temporal traffic prediction task.

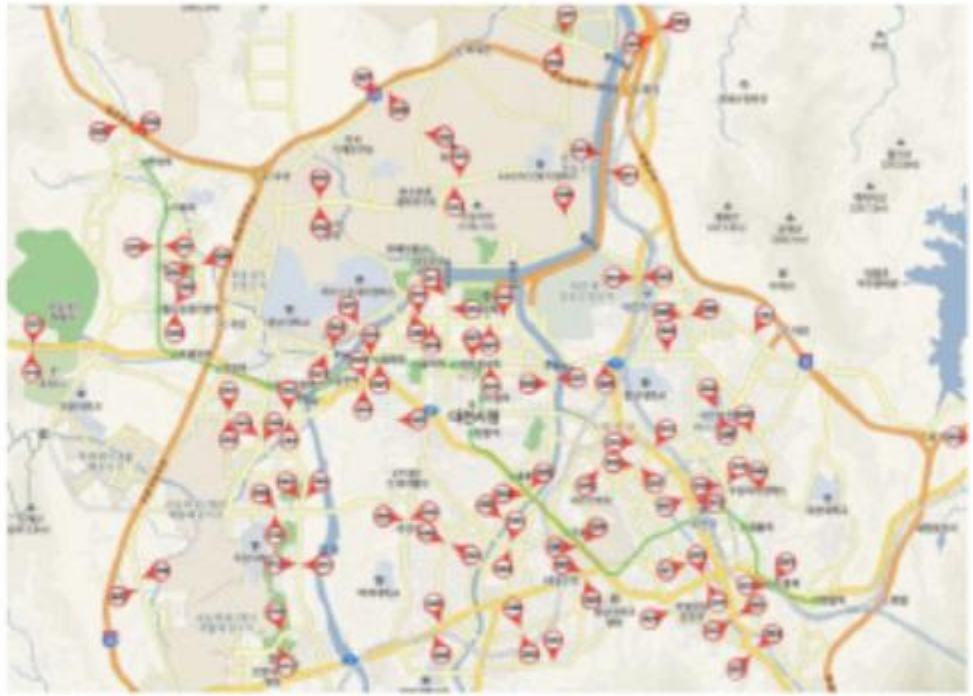
Datasets	Characteristic	Time interval (min)	Source (https://github.com)
PeMS-BAY	Speed	5	/liyaguang/DCRNN (2018)
METR-LA	Speed	5	/liyaguang/DCRNN (2018)
LOOP	Speed	5	/zhiyongc/Seattle-Loop-Data (2018)
PeMSD4,8	Volume	5	/Davidham3/ASTGCN (2018)
Q-Traffic	Speed	15	/JingqingZ/BaiduTraffic (2018)
NYC Taxi	Demand	30	/toddwschneider/nyc-taxi-data (2019)
PeMSD3,7	Volume	5	/Davidham3/STSGCN (2020)

UVDS: Urban VDS Dataset

- Tra±c data have been collected from VDS
  - ✓speed, traffic volume, occupancy, and vehicle types.



(a) METR-LA



(b) UVDS

- For graph construction, UVDS based on the geometric distances between sensors (N=104)

$$W_{i,j} = \begin{cases} \exp\left(-\frac{d^2_{v_i,v_j}}{\sigma^2}\right), & \text{if } \exp\left(-\frac{d^2_{v_i,v_j}}{\sigma^2}\right) \geq \beta, \\ 0, & \text{otherwise,} \end{cases}$$

$$d_{v_i,v_j} = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_j - \phi_i}{2}\right) + \cos(\phi_i)\cos(\phi_j)\sin^2\left(\frac{\varphi_j - \varphi_i}{2}\right)}\right)$$

Results on UVDS dataset

- Bolded texts are best results

Step	Metrics	MTGNN	STAWnet	DSTWN
1	MAE	3.4671	3.4535	**3.4501**
	RMSE	5.2665	**5.2494**	5.2517
	MAPE	0.0746	0.0740	**0.0736**
3	MAE	**3.6504**	3.6609	3.6577
	RMSE	**5.5896**	5.6015	5.6105
	MAPE	0.0800	0.0804	**0.0799**
6	MAE	**3.7740**	3.8000	3.7903
	RMSE	**5.8132**	5.8491	5.8417
	MAPE	**0.0832**	0.0846	0.0840
9	MAE	3.8629	3.8753	**3.8556**
	RMSE	5.9516	5.9681	**5.9507**
	MAPE	**0.0856**	0.0865	0.0859
12	MAE	3.9589	3.9469	**3.9144**
	RMSE	6.0799	6.0687	**6.0362**
	MAPE	0.0887	0.0886	**0.0872**
Training (s/epoch)		16.7419	23.1326	30.4803
Inference (s/epoch)		0.4241	0.5603	0.7769

OPEN ACCESS
Vietnam Journal of Computer Science
(2022)

DOI: 10.1142/S2196888822500324

Transformer Based Traffic flow prediction Research

KISTI SINCE 1962

- ❖ 대전시 데이터웨어하우스에서 차량검지기(VDS)와 RSE 데이터 수집 및 분석함
  - ✓KISTI 정문 앞 도로 차량검지기(VDS 17데이터)는 오전과 출근과 오후 퇴근에 일부 속도가 떨어지는 경향이 있음.
  - ✓대학로 인근 RSE 데이터는 전체적으로 속도가 너무 낮게 측정되었음.

- ❖ (딥러닝 기반 교통 흐름 예측)
  - ✓RNN 기반 LSTM을 양방향 장단기로 교통 흐름 예측하였고 교통 혼잡은 없는 것으로 예측됨
  - ✓Transformer_VDS 모델 개발로 장기 교통흐름 예측 정확도가 향상됨
  - ✓Spatial-Temporal Graph 데이터 기반 Dynamic Spatial Transformer WaveNet 모델 개발 및 성능 테스트 함

2. Transformer 모델로 예측

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (1)

```python
import tensorflow as tf
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import pandas as pd
import numpy as np
df = pd.read_csv('./input/daejeon_vds16.csv')
df.head()
df.set_index('Date', inplace=True)

spd=df.reset_index()['Speed']
windows = 60
X, y = [], []
for i in range(len(spd)-windows-1):
    X.append(spd[i:(i+windows)])
    y.append(spd[(i+windows)])
```

```python
X = np.array(X)
y = np.array(y)
X.shape
X[:3]
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (2)

```
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y,
28                                                    test_size=0.20, shuffle=False)
29 print(X_train.shape, y_train.shape)
30 print(X_test.shape,  y_test.shape)
31
32 X_train = np.array(X_train).reshape(X_train.shape[0], X_train.shape[1], 1)
33 X_test = np.array(X_test).reshape(X_test.shape[0], X_test.shape[1], 1)
34 print(X_train.shape, X_test.shape)
35
36 X_train_ = X_train.reshape(X_train.shape[0],X_train.shape[1],1)
37 X_test_ = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
38 print(X_train_.shape, X_test_.shape)
```

```
(6402, 60) (6402,)
(1601, 60) (1601,)
(6402, 60, 1) (1601, 60, 1)
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (3)

```
39
40 ### G. Transformer 모델을 적용해보자
41 from keras import backend as K
42 from tensorflow import keras
43 from tensorflow.keras import layers
44
45 def transformer_encoder(inputs, head_size, num_heads, ff_dim, dropout=0):
46     x = layers.LayerNormalization(epsilon=1e-6)(inputs)
47     x = layers.MultiHeadAttention(key_dim=head_size, num_heads=num_heads,
48                                   dropout=dropout)(x, x)
49     x = layers.Dropout(dropout)(x)
50     res = x + inputs
51     x = layers.LayerNormalization(epsilon=1e-6)(res)
52     x = layers.Conv1D(filters=ff_dim, kernel_size=1, activation = "relu")(x)
53     x = layers.Dropout(dropout)(x)
54     x = layers.Conv1D(filters=inputs.shape[-1], kernel_size=1)(x)
55     return x + res
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (4)

```
57 def build_model(input_shape, head_size, num_heads, ff_dim, num_transformer_blocks,
58                 mlp_units, dropout=0, mlp_dropout=0):
59
60     inputs = keras.Input(shape=input_shape)
61     x = inputs
62     for _ in range(num_transformer_blocks):  # This is what stacks our transformer blocks
63         x = transformer_encoder(x, head_size, num_heads, ff_dim, dropout)
64
65     x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
66     for dim in mlp_units:
67         x = layers.Dense(dim, activation="elu")(x)
68         x = layers.Dropout(mlp_dropout)(x)
69
70     outputs = layers.Dense(1, activation="linear")(x) #this is a pass-through
71     return keras.Model(inputs, outputs)
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (5)

```
73 def lr_scheduler(epoch, lr, warmup_epochs=30, decay_epochs=100,
74                   initial_lr=1e-6, base_lr=1e-3, min_lr=5e-5):
75     if epoch <= warmup_epochs:
76         pct = epoch / warmup_epochs
77         return ((base_lr - initial_lr) * pct) + initial_lr
78 
79     if epoch > warmup_epochs and epoch < warmup_epochs+decay_epochs:
80         pct = 1 - ((epoch - warmup_epochs) / decay_epochs)
81         return ((base_lr - min_lr) * pct) + min_lr
82 
83     return min_lr
84 
85 callbacks = [ tf.keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True),
86               tf.keras.callbacks.LearningRateScheduler(lr_scheduler)       ]
87 
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (6)

```
input_shape = X_train_.shape[1:]
print(input_shape)
model = build_model(input_shape, head_size=46,  num_heads=60,  ff_dim=55,
                    num_transformer_blocks=5, mlp_units=[64], mlp_dropout=0.4, dropout=0.14, )
model.compile( loss="mean_squared_error",
               optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
               metrics=["mean_squared_error"])
model.summary()
history = model.fit( X_train_, y_train, validation_split=0.2, epochs=30, batch_size=20, callbacks=callbacks)

import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='best')
plt.show()
```

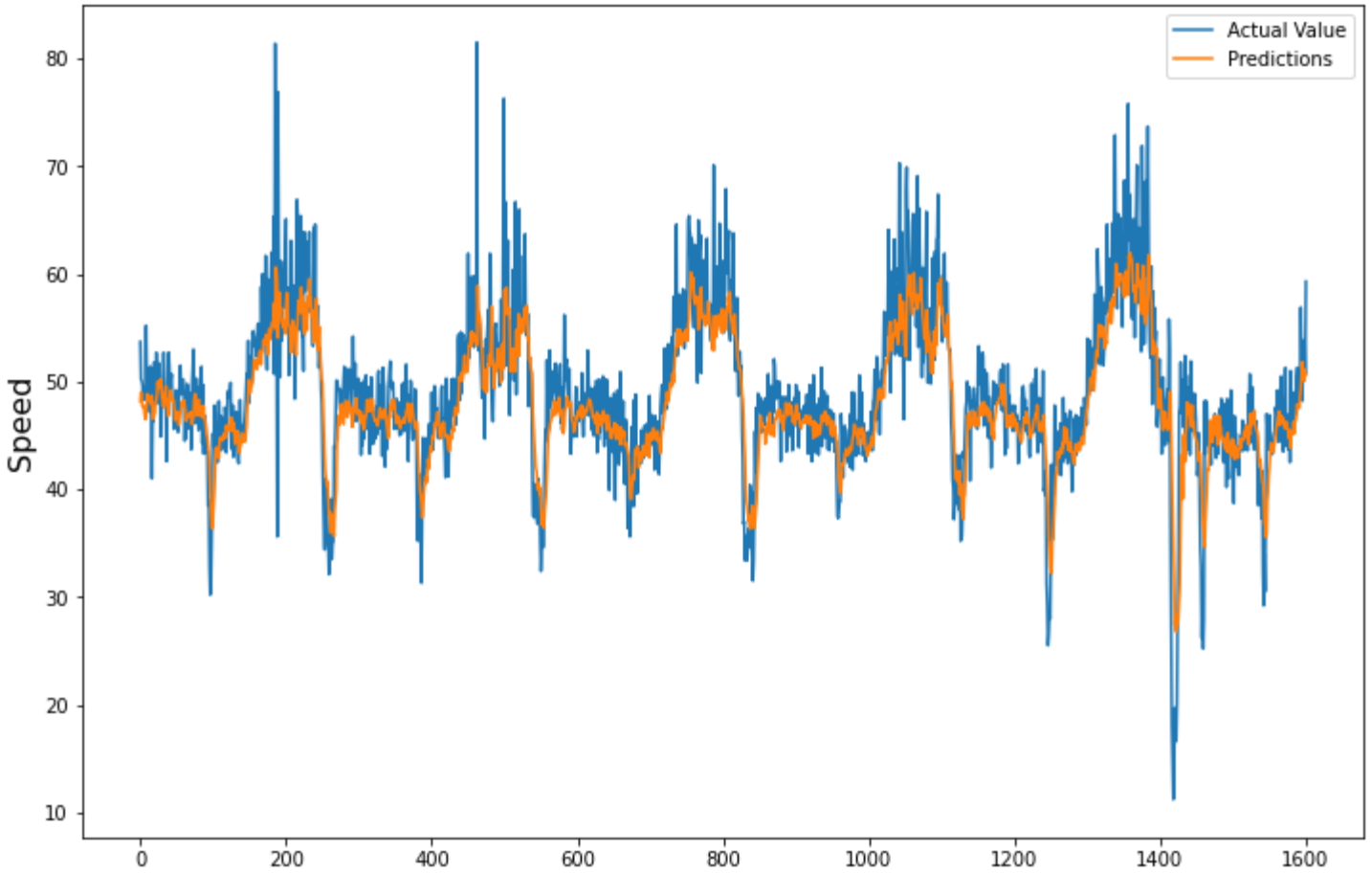
(day3-3) VDS 속도를 Transformer 모델로 예측하기 (7)

```
106 y_pred = model.predict(X_test_)
107 pred_df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
108 pred_df.head()
109 Acc=[]
110 from sklearn.metrics import r2_score
111 print("Accuracy score of the predictions: {0}".format(r2_score(y_test, y_pred)))
112 Acc.append(r2_score(y_test, y_pred))
113 plt.figure(figsize=(12,8))
114 plt.ylabel('Speed', fontsize=16)
115 plt.plot(pred_df)
116 plt.legend(['Actual Value', 'Predictions'])
117 plt.show()
```

(day3-3) VDS 속도를 Transformer 모델로 예측하기 (8)

❖ 실행결과

✓windows=60



KISTI SINCE 1962

2022 TRUST KISTI

Korea Institute of Science and Technology Information