KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information

2023

스마트교통 빅데이터 분석

2023.6.19.

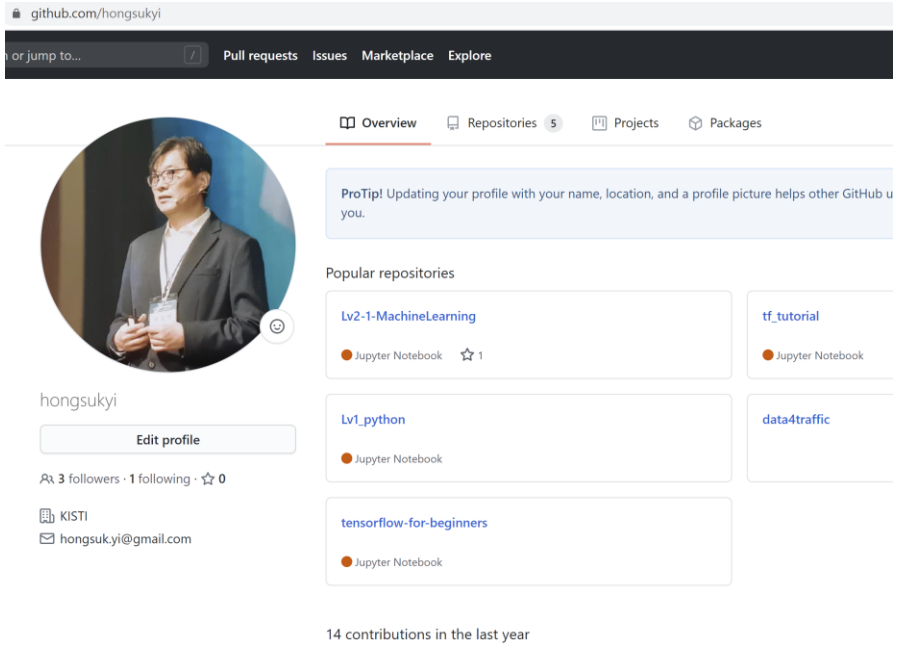
이 홍 석 (hsyi@kisti.re.kr)

강사소개

- 성명 : 이홍석
- 소속
  - ✓ KISTI 데이터기반문제해결연구단 책임연구원
  - ✓ 과학기술연합대학원(UST-KISTI) 응용AI전공 교수
- 연구내용
  - ✓ 딥러닝기반 도심지 교통혼잡 예측 (2018~2021)
  - ✓ 지능형 인프라 기술 연구 (2018~2019)
  - ✓ 도심지 교통예측을 위한 트랜스포머 모델 연구(2022~2025)
- 컴퓨팅 기술
  - ✓ GPU 기반 가속컴퓨팅 기술 : CUDA, OpenCL
  - ✓ 슈퍼컴퓨팅 병렬처리 기술 : MPI, OpenMP, ManyCore Computing
  - ✓ Deep Learning 기술 : Tensorflow, PyTorch, Keras, Theano

강사소개

- 교육 경험: 머신러닝, 딥러닝, 인공지능, 빅데이터, MPI/OpenMP, CUDA, OpenCL 등
  - ✓ 딥러닝 워크숍 개최 (2015)
  - ✓ 한국교통학회 텐서플로우 튜토리얼 강좌 (2016~2018)
  - ✓ 교통학회/KIRD/KISTI 과학데이터스쿨 강의 진행 (2018~2022)
  - ✓ 교통 빅데이터 교육 (2022)
  - ✓ UST 인공지능 특론 (전공필수) (2022)
  - ✓ UST 인공지능 이해 (교양) (2021~2022)
  - ✓ UST 기계학습의 이해, 인공지능 기초 (2019~2022)
- 참고자료
  - ✓ 인공지능, 딥러닝 강의 자료 모음 (github.com/hongsuk.yi)



강의 진행

- 수업 운영
  - ✓ 이론
  - ✓ 실습(예제)
- 강의자료
  - ✓ https://github.com/hongsukyi/Traffic_Data_ML
- 실습관련
  - ✓ Google Colab 활용
    - 브라우저에서 파이션을 작성하고 실행할 수 있음
    - 실행환경 구성 필요 없음
    - GPU 무료 액세스
    - 간편한 공유
  - ✓ 실습 시 개인별 Google 계정 필요

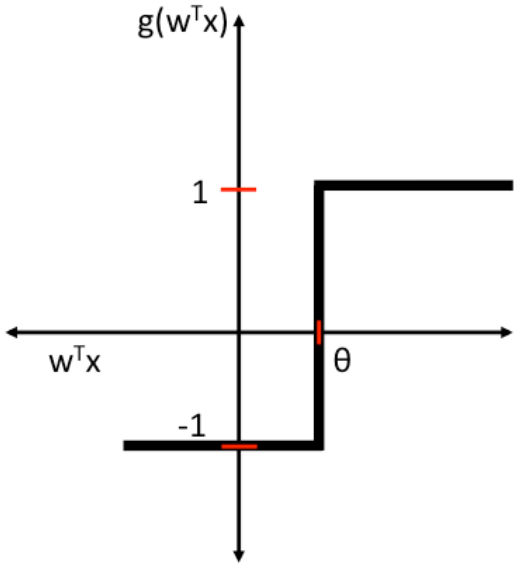
학습 목표

- 1일차 : 분류 모델
  - ✓ 교통 빅데이터의 이해 및 전처리 (2H)
  - ✓ 머신러닝을 이용한 교통 빅데이터 분류 (2H)
  - ✓ Deep Neural Network을 이용한 교통 흐름 분류 (2H)
- 2일차 : 회귀 및 예측
  - ✓ 회귀 소개와 MLP(DNN) 회귀 (2H)
  - ✓ RNN 소개와 한스텝 예측 모델 (2H)
  - ✓ SimpleRNN, LSTM, GRU 모델을 이용한 예측 (2H)
- 3일차 : 예측
  - ✓ CNN 소개와 교통흐름 예측을 위한 Conv1D 모델 (2H)
  - ✓ (예측) 멀티스템 예측 모델 : DNN, RNN, LSTM, CNN (2H)
  - ✓ 프로젝트 (2H)

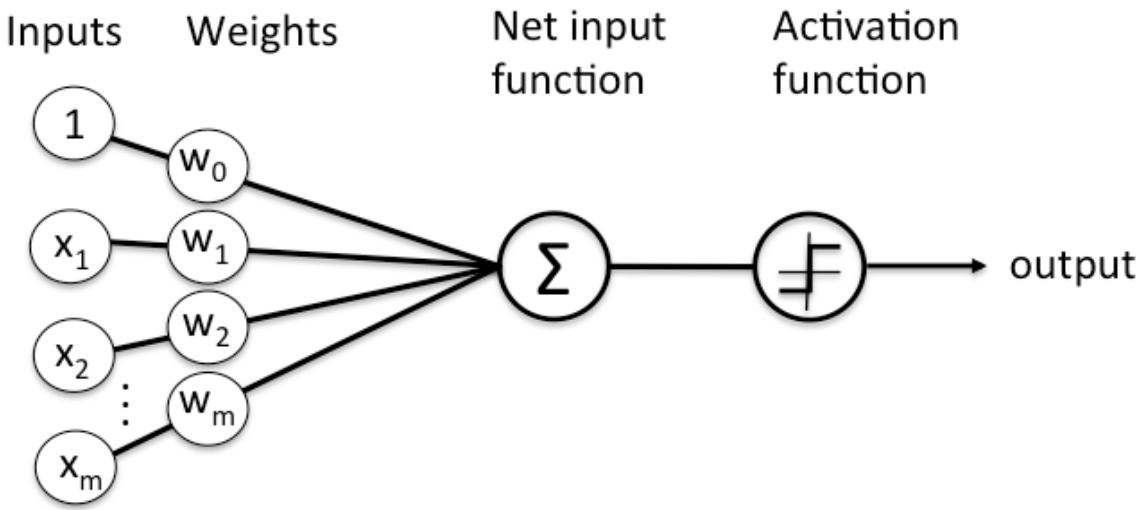
1) 인공지능 소개와 역사

The perceptron (1958)

- Rosenblatt's perceptron
  - ✓ 활성함수는 TLU(계단함수)



Unit step function.



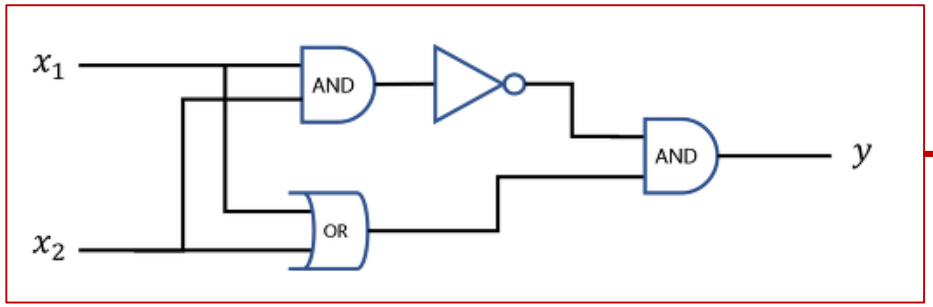
Schematic of Rosenblatt's perceptron.

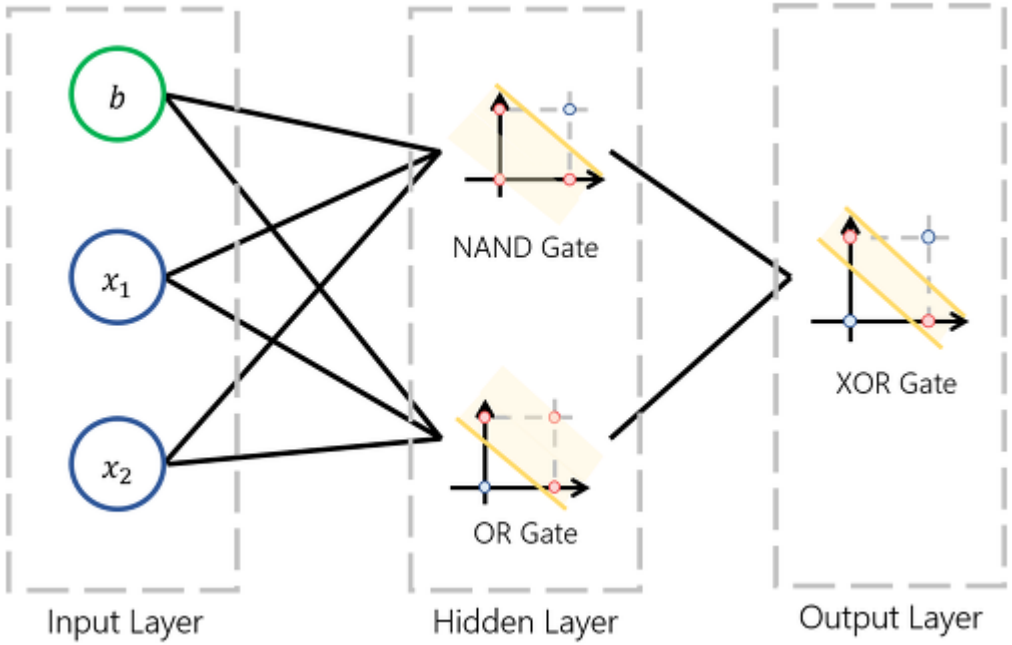
(source) https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

MLP : Multilayer Perceptron

- MLP로 'XOR' 문제 해결
  - ✓ XOR 게이트는 'AND'와 'OR', 'NOT' 게이트의 조합
  - ✓ Perceptron으논 'AND'나 'OR' 게이트 만들 수 있다.
  - ✓ 여러 개의 Perceptron을 조합하면 'XOR' 게이트 만들 수 있다.

XOR = !AND && OR

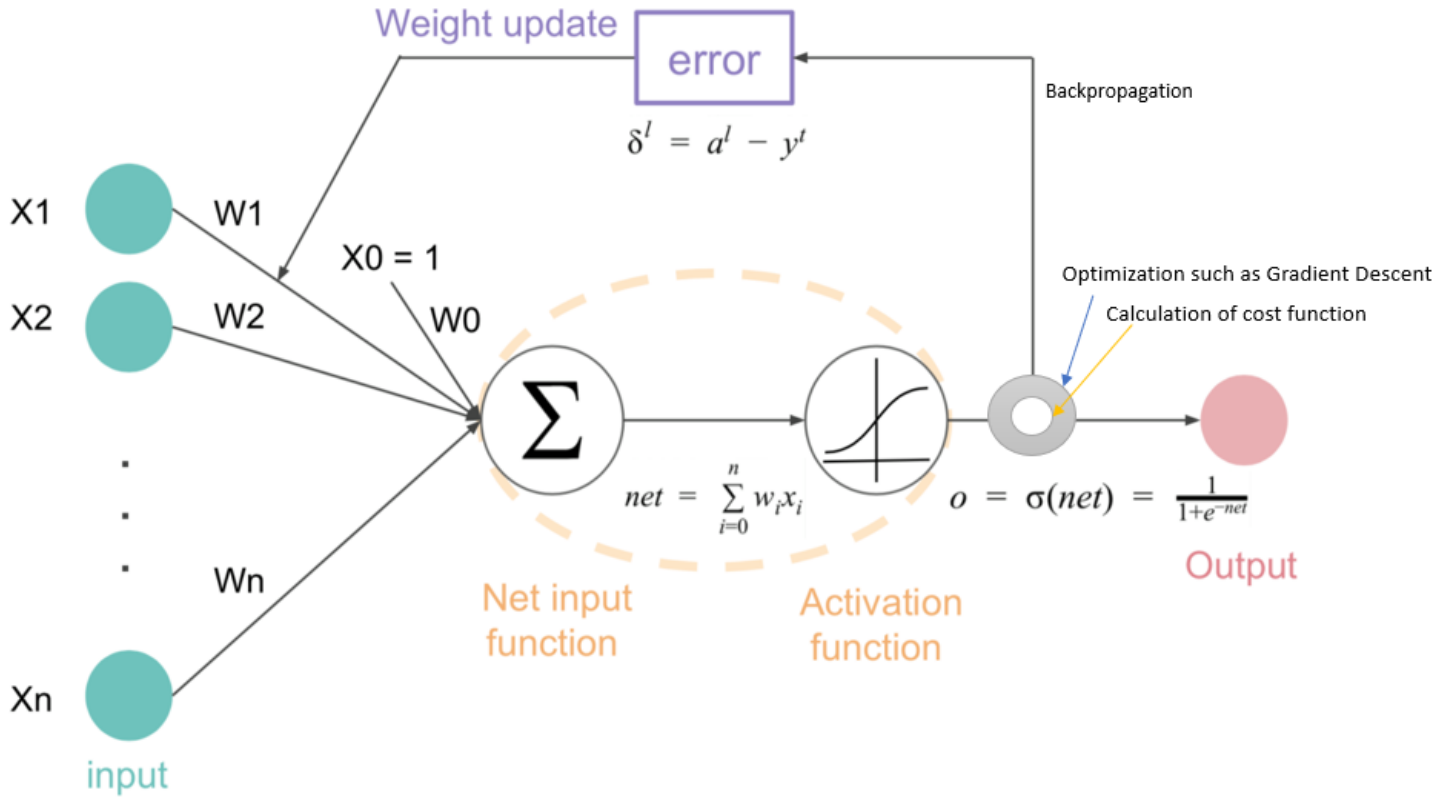




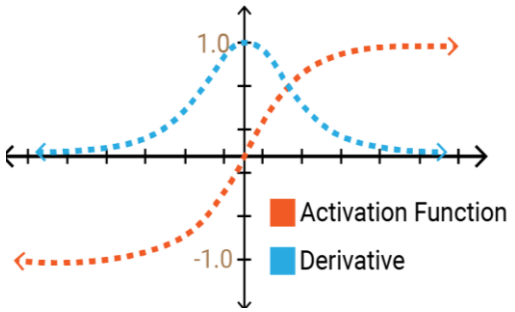
(source) https://gomguard.tistory.com/178

Backpropagation로 AI 다시 봄이 시작된다

❖ 역전파알고리즘 (Geoffrey E. Hinton, 1986)



활성함수로 TLU(계단함수) 대신
미분가능한
Sigmoid 함수 사용

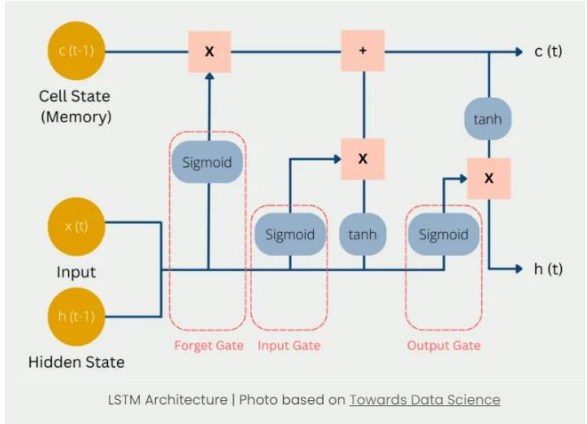


(source) https://datascience.stackexchange.com/questions/44703/how-does-gradient-descent-and-backpropagation-work-together

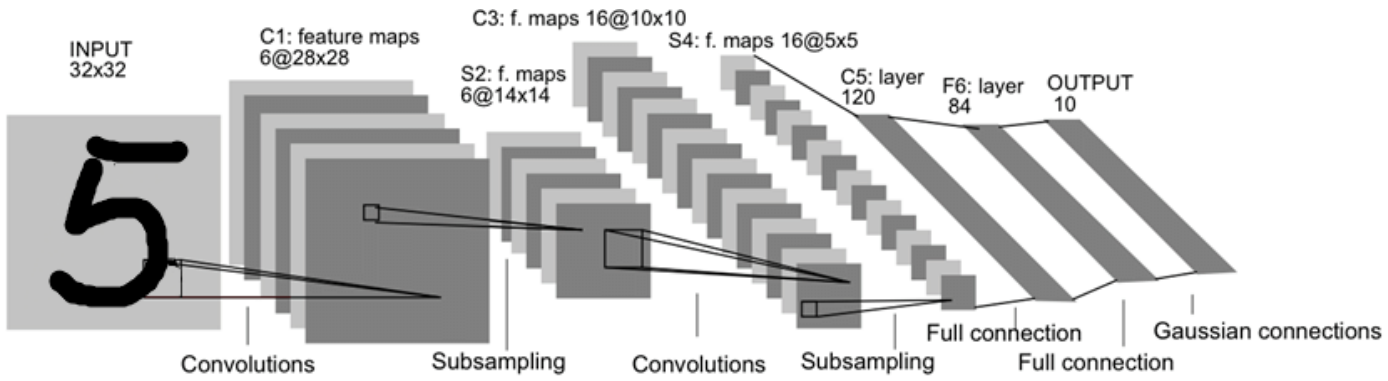
인공지능 2차 겨울 극복기 (1993~2006)

❖ 문제점 및 해결

- ✓ ReLU로 Deep Learning 가능
- ✓ MLP의 오버피팅(Overfitting) 문제
  - CNN의 가중치 공유
- ✓ RNN의 VGP 문제는
  - LSTM (Long-Short Term Memory)
- ✓ GPU 등 가속기 등장



(source) https://databasecamp.de/en/ml/lstms



LeNet (가중치 공유)

(source) https://www.researchgate.net/figure/The-architecture-of-LeNet-5-23-a-CNN-used-for-digits-recognition-for-the-MNIST-dataset_fig2_321665783

Deep Learning in Neural Networks : ReLU (2010~)

- ReLU 선형 활성함수가 중요한 기여를 한다.
  - ✓ 첫 사용은 이미 1969년 (Fukushima)

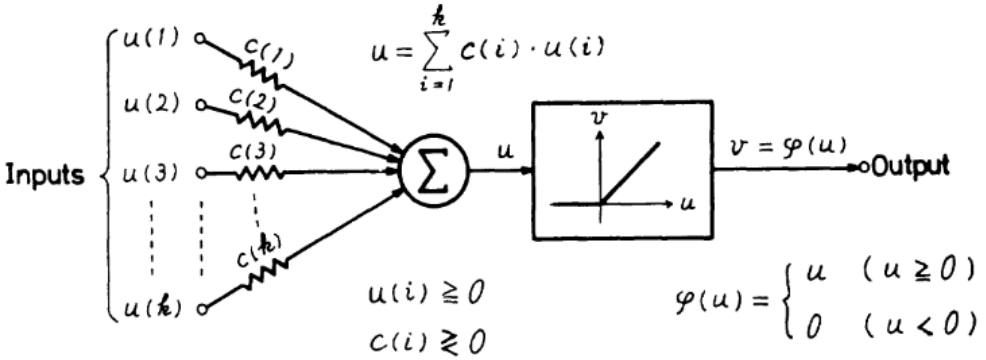
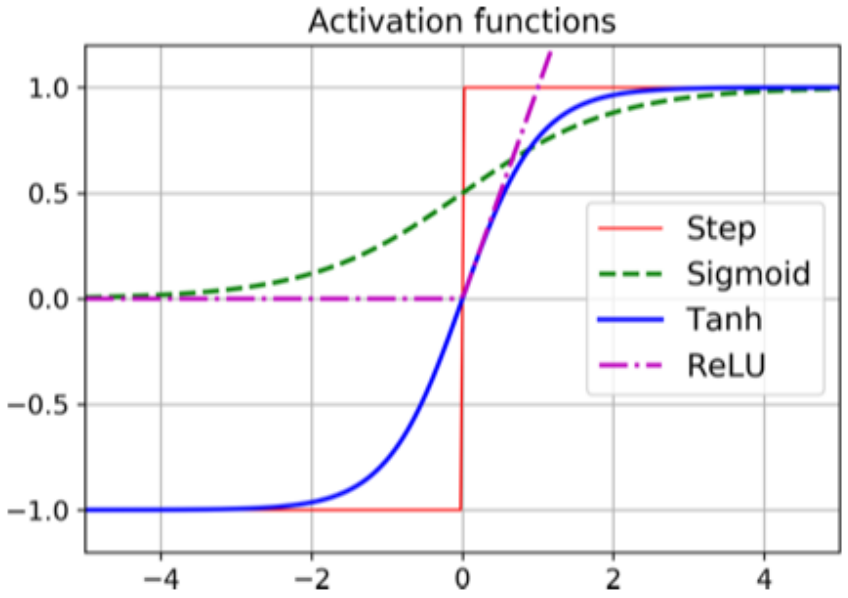


Fig. 3. Analog threshold element employed in the feature extractor network.

  - ✓ 최근은 "Rectified Linear Units improve RBM' (G.Hinton, 2010년)

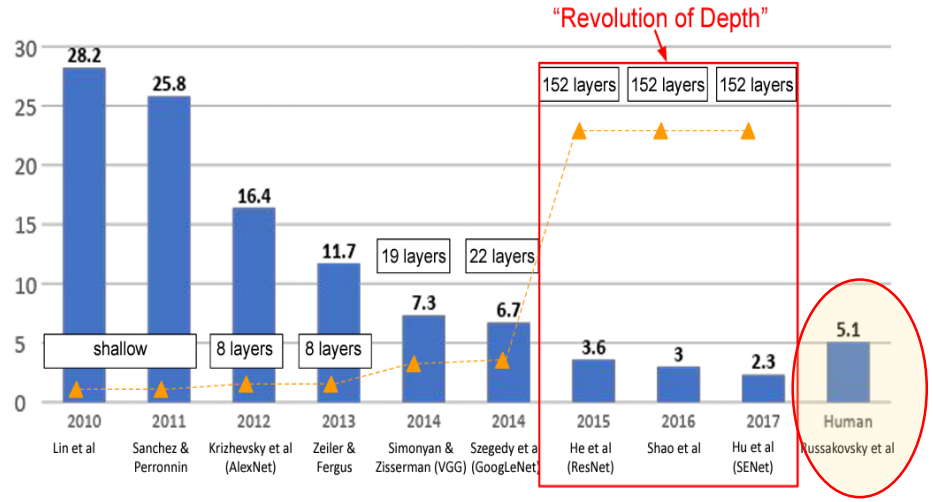


(source) https://stats.stackexchange.com/questions/447674/when-was-the-relu-function-first-used-in-a-neural-network

인공지능 성공 사례 : 3가지

1) ILSVRC 이미지 분류 모델 (CNN)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



2) AlphaGo (강화학습)

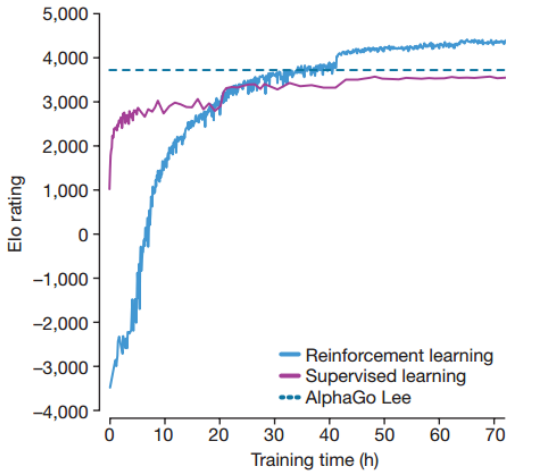
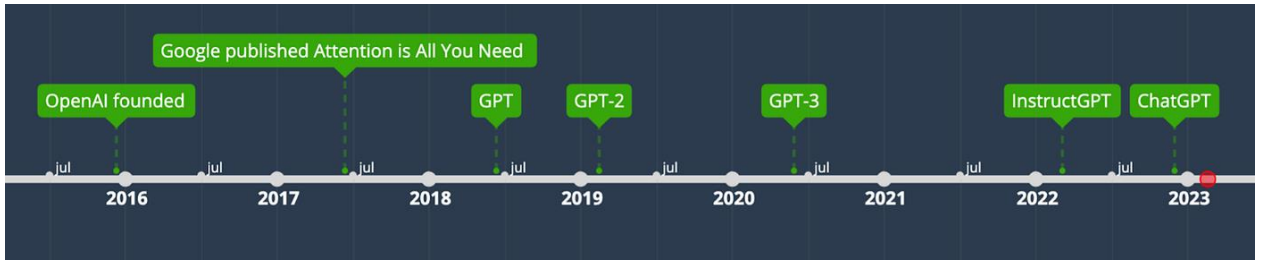


Figure 3 | Empirical evaluation of AlphaGo Zero. a

(Source) Mastering the game of Go without human knowledge , David Silver, et al. Nature(2017)

3) ChatGPT (텍스트 생성)

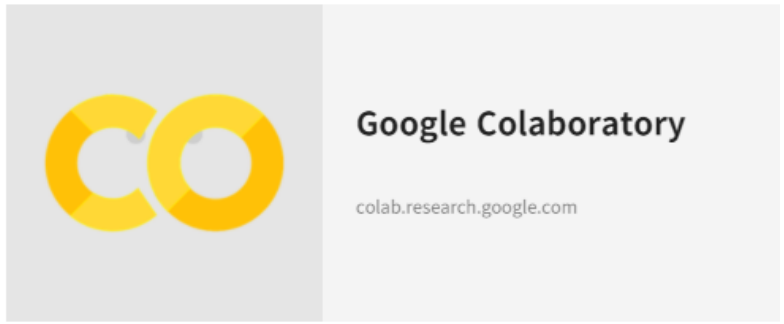


(source) https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Google Colab 스펙

- 코랩이란?
  - ✓ 클라우드에서 실행되는 무료 Jupyter 노트 환경
- **Google Colaboratory 접속하기**
  - ✓ https://colab.research.google.com 접속
- Google Colab 스펙
  - ✓ CPU : 제온
  - ✓ Memory : 13GB
  - ✓ HDD : 320GB
  - ✓ GPU : NVIDIA Tesla K80

https://colab.research.google.com/

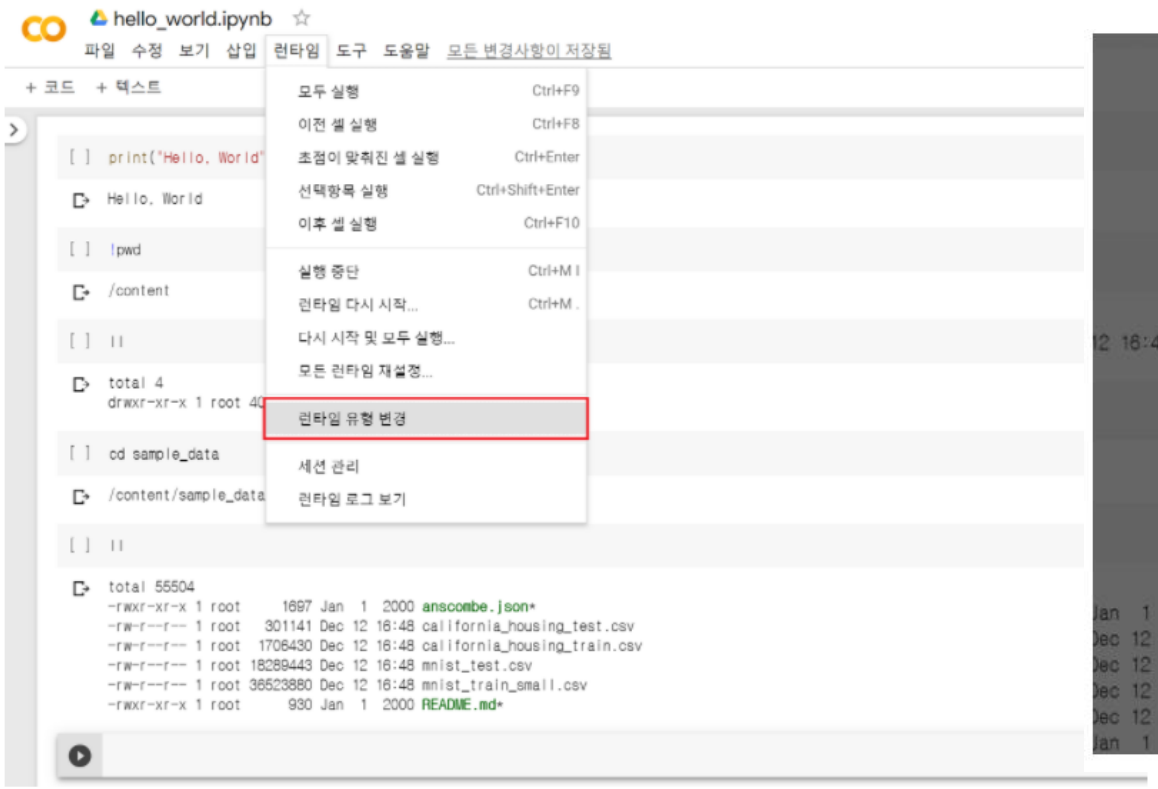


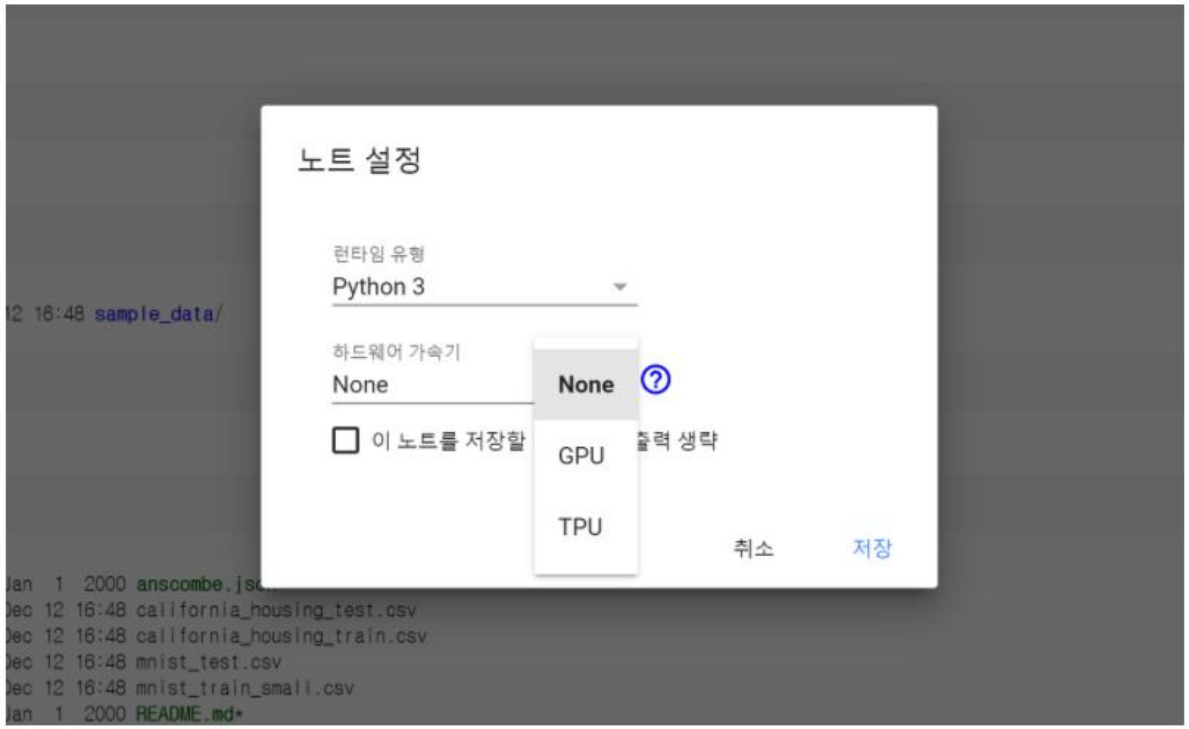
런타임 유형 GPU, TPU 설정하기

무료답지 않게 엄청난 서비스가 무료입니다.

GPU, TPU 등을 지원하기 때문에 요즘 핫한 Tensorflow등을 동작할 수 있습니다.

Colab의 GPU 정보 확인 방법:
!nvidia-smi





2) 교통데이터 및 AI 모델 소개

도시 교통 문제 : 지속 가능성

- 도시 교통 문제의 원인
  - ✓ 교통혼잡은 온실가스, 미세먼지 원인
  - ✓ 도시문제는 지속 가능성에 대한 글로벌한 도전
- 도시문제 해결을 위한 핵심 키워드는 스마트 시티(Smart city)

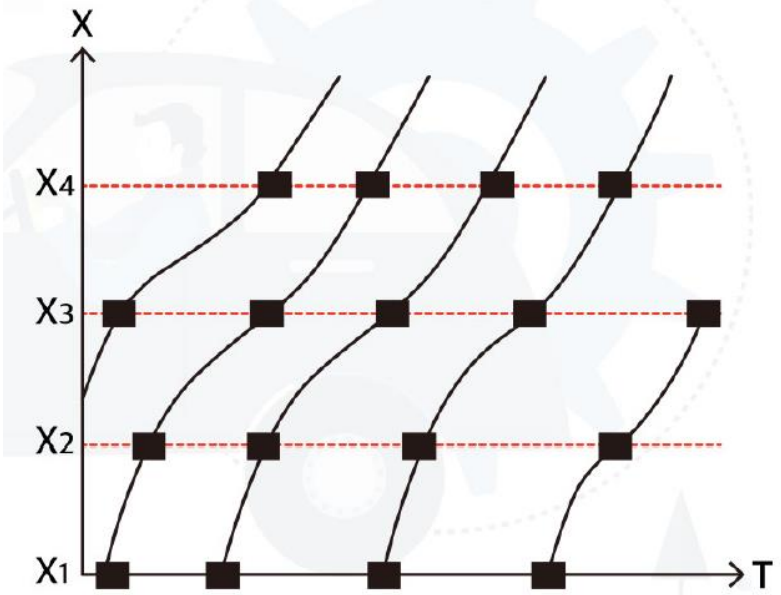
교통 데이터 관측

❖ 고정위치 관측

- ✓ 도로상의 고정된 위치에서 지나가는 차량들을 관측
- ✓ 고정된 시각에 일정 구간의 교통 상태를 관측
- ✓ 항공 사진 촬영

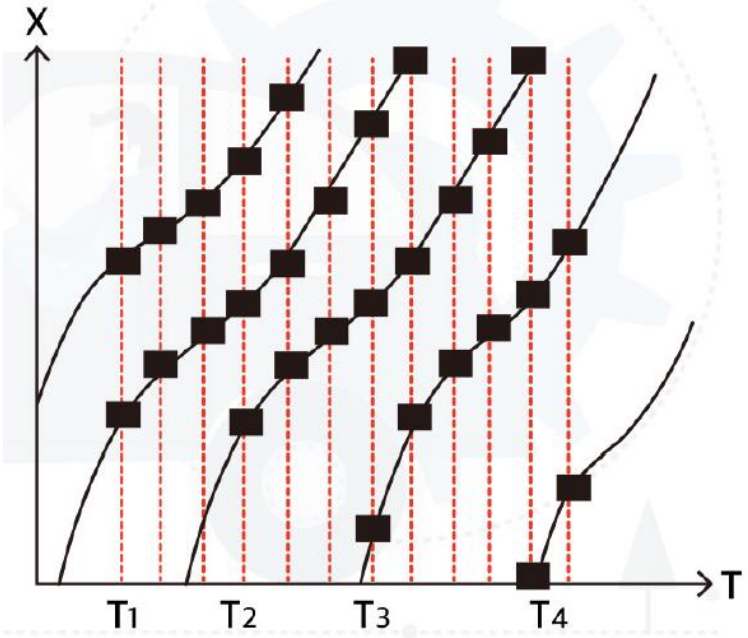
❖ 관측가능한 데이터는

- ✓ 통과 차량수, 차량 사이의 시간 간격, 차량 속도



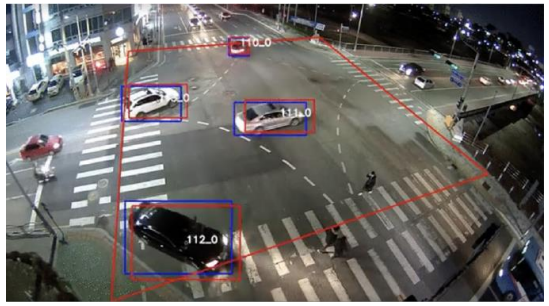
● 연속 시간 관측

- ✓ 일정 구간내의 교통상태를 연속된 시간동안 관측
- ✓ 교통 카메라
- ✓ 관측데이터



교통 센서: 영상 검지기

영상 검지기



RSU(Road Side Unit)

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

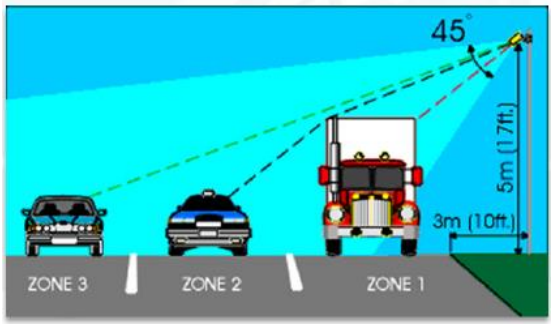
VDS(Vehiche Detection System)

교통량
점유률
속도
시간(1분,5분)

루프 검지기

- 교통량 산정
- 도로 용량 분석
- 차량 통과 속도 감시

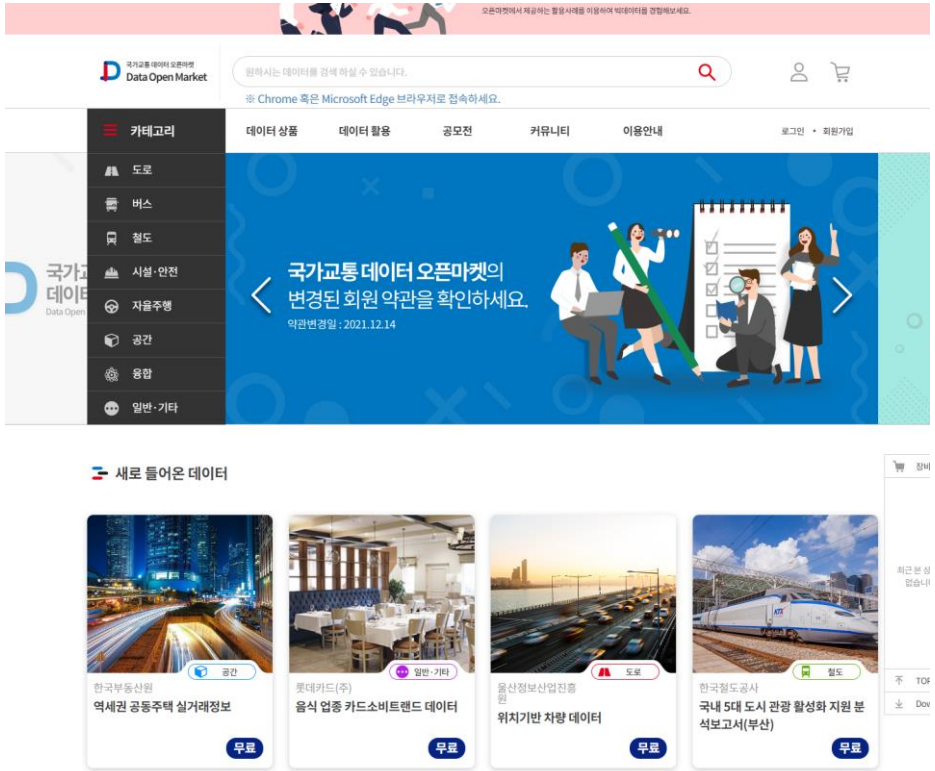
교통 레이다



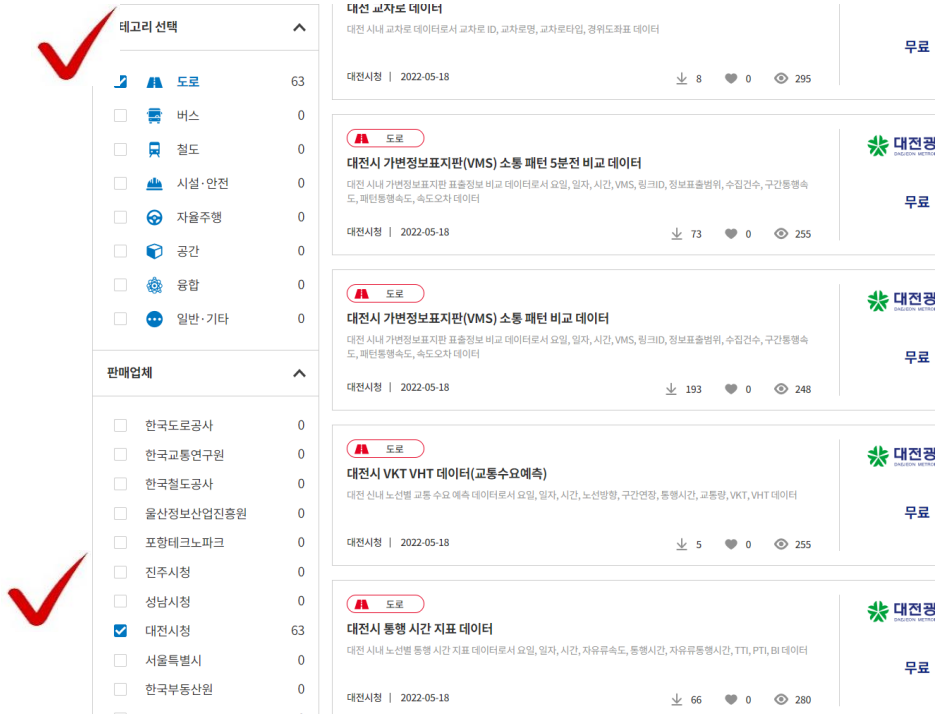
교통 데이터 다운로드

국가교통데이터 오픈마켓 (로그인 필수로 하세요)

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



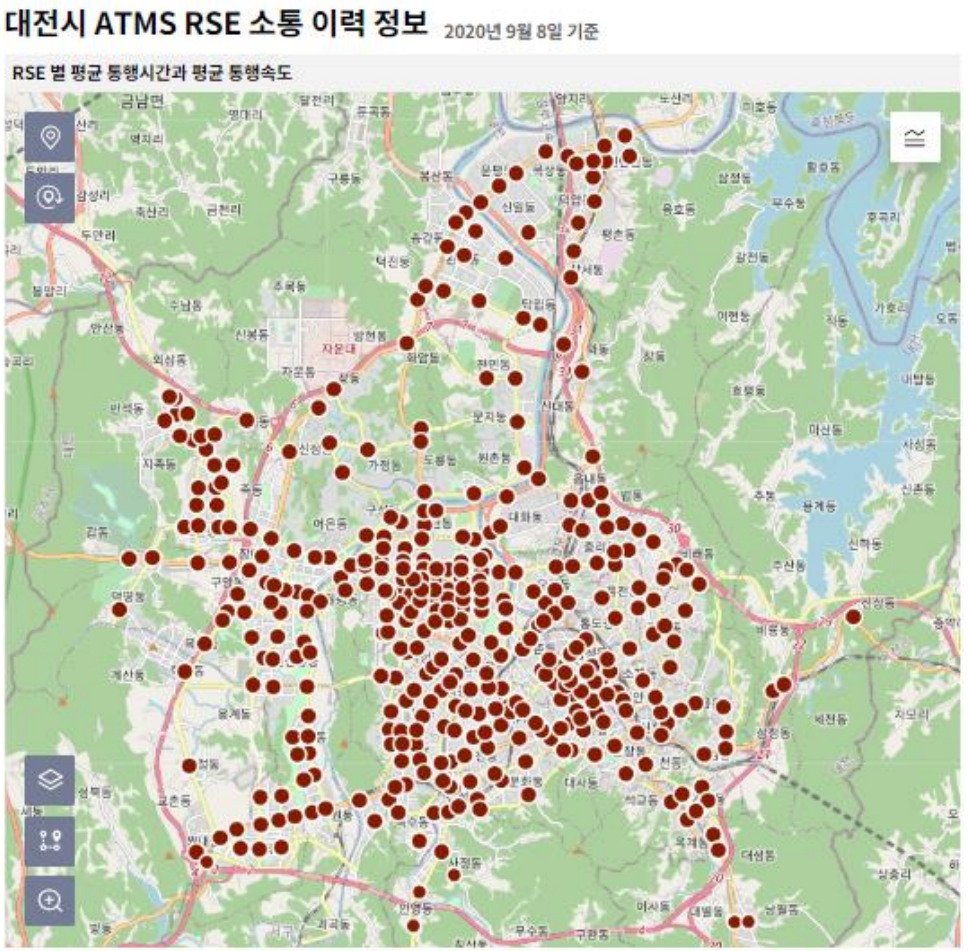
(예제) 도로/대전시청을 선택



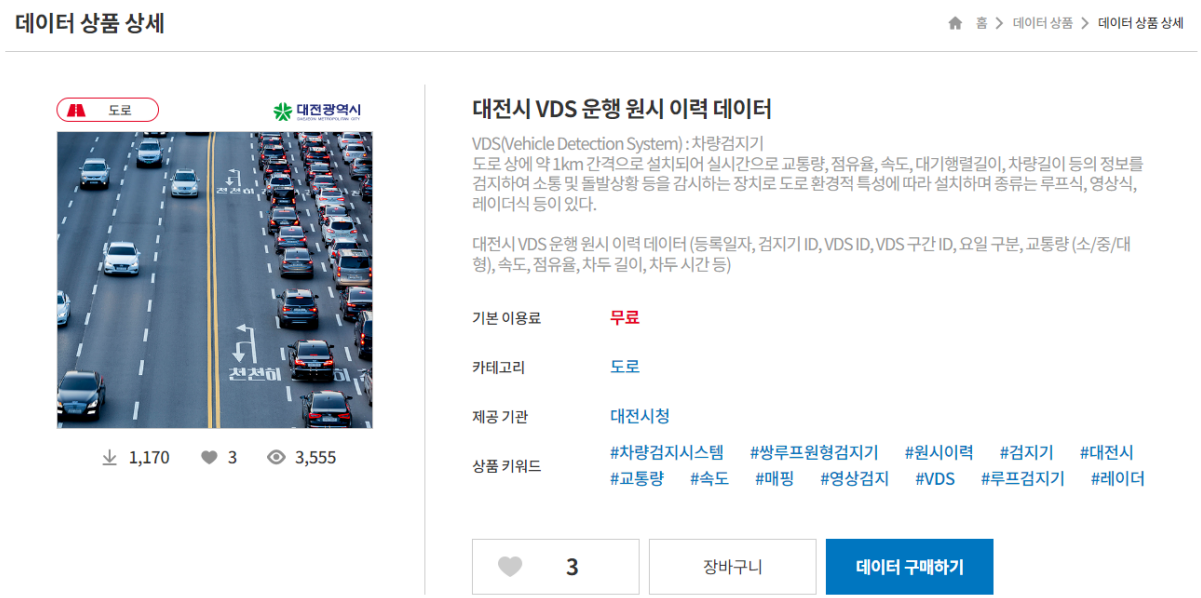
RSE(Road-Side Equipment, 노변장치) 센서 위치 : 대전시

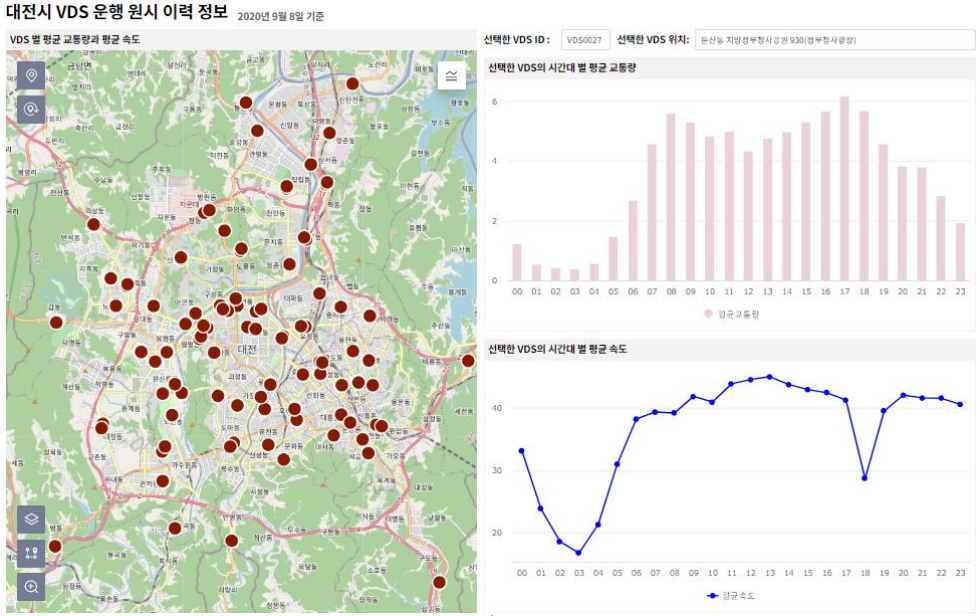
❖ 첨단 교통량 관리 시스템

- ✓ (ATMS, Advanced Transportation Management System)
  - 실시간 교통상황 정보를 토대로 도로와 같은 교통시설의 이용률을 극대화하기 위한 교통관제체계
- ✓ RSE란
  - 교차로나 차로변에 위치한 CCTV 지주 등 에 설치되어 차량의 운행정보를 수집하는 기기
- ✓ 대전시 노변장치(RSE) 정보 5분 단위 데이터
  - 일자 ID, 시 ID, 분 ID, 링크 ID, RSE 통행 속도, RSE 통행 시간 등
- ✓ RSE 위치 및 각 RSE 별 평균 통행시간/통행속도



대전시 VDS 운행 원시 이력 데이터





VDS 데이터 상세 설명

교통량(q), 속도(v), 밀도(k)와의 관계

- 연속류의 대표적인 효과척도라 할 수 있는 교통량, 속도, 밀도의 상관관계
- 교통량(Q)
  - ✓ 단위시간(일, 시간, 15분 등)에 어떤 지점을 통과하는 차량의 총대수(대)
- 평균통행속도(km/hr)
  - ✓ 일정구간거리를 통행시간으로 나눈 평균값
- 밀도(승용차/km/차로)
  - ✓ 어느 시간에 단위구간 도로상에 있는 차량대수 (대/km/시간)

Q = V × K	Q : 교통량 V : 속 도 K : 밀 도

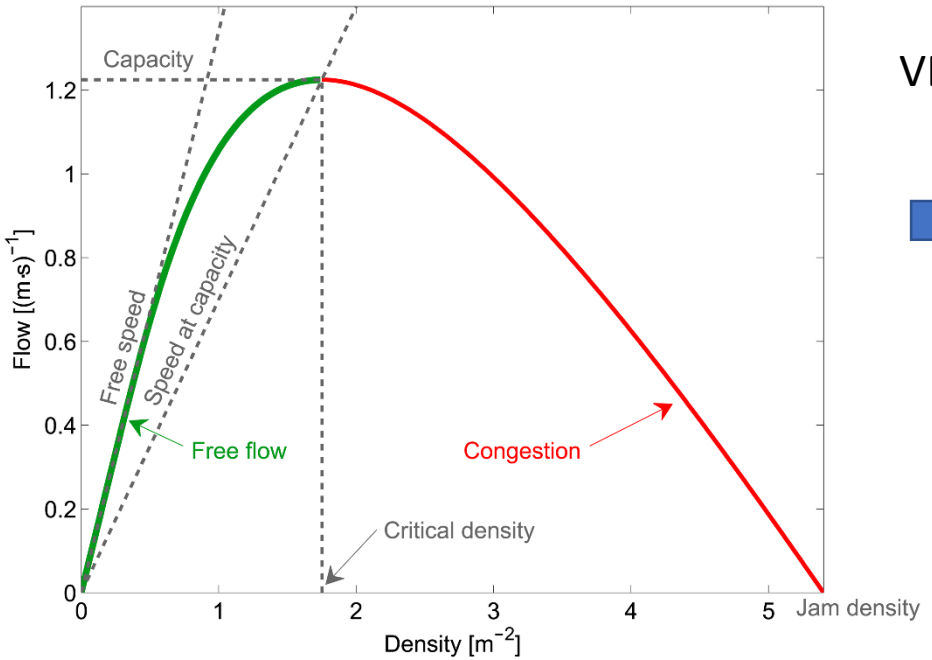
- 교통량이 일정할때 밀도가 높으면 속도저하 발생

- 3요소 상호간의 작용에 의해 각 요소의 성격이 규명된다는 가정하에 접근

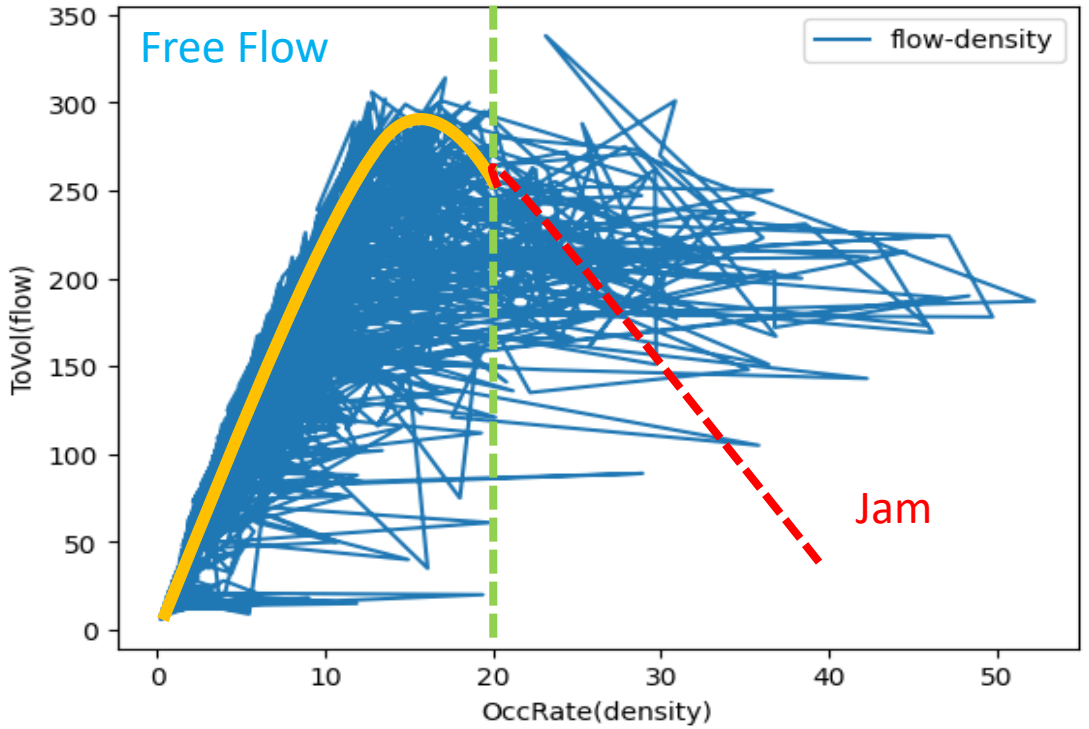
https://moneyjjang.tistory.com/305#gsc.tab=0

교통량-밀도(점유율)

- 교통량이 많으면 밀도 증가 (자유류)
- 교통량이 용량을 초과하면 밀도는 증가하지만 교통량은 감소(혼잡류)



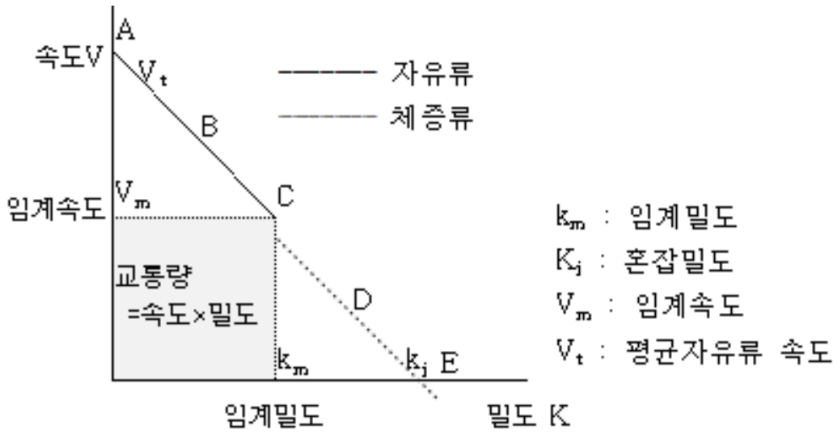
VDS16



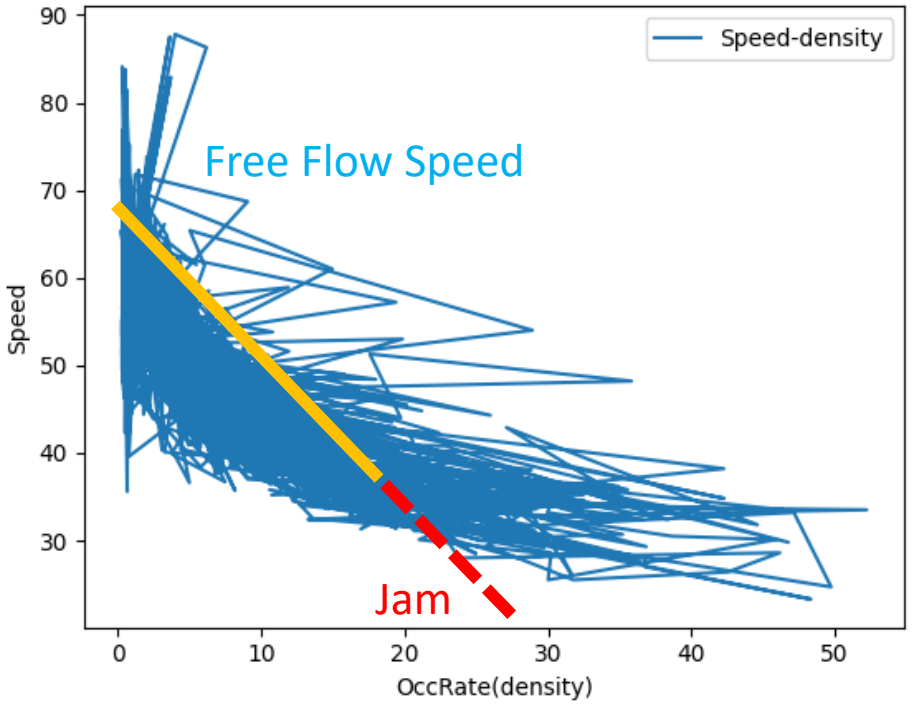
참고자료: https://journals.plos.org/plosone/article/figure?id=10.1371/journal.pone.0208496.g001

속도-밀도(점유율)

- 밀도가 증가하면 속도 저하
- 도로의 최대교통류율은 도로의 용량이 되며
  이때 밀도가 임계밀도, 이때 속도가 임계속도



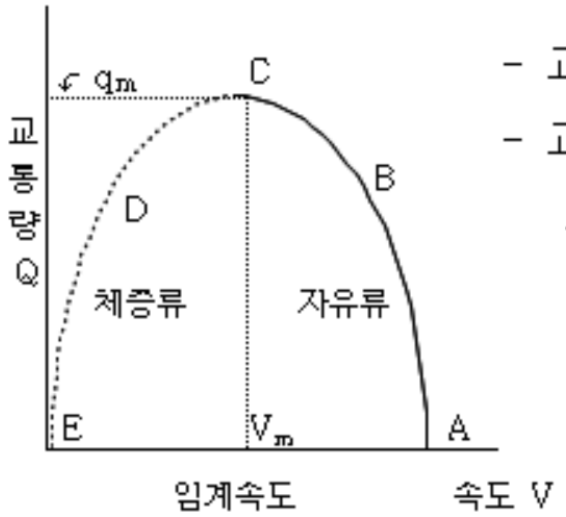
VDS16



참고문서: https://rashidfaridi.com/2018/12/11/concept-of-traffic-flow-diagram/

Speed-flow

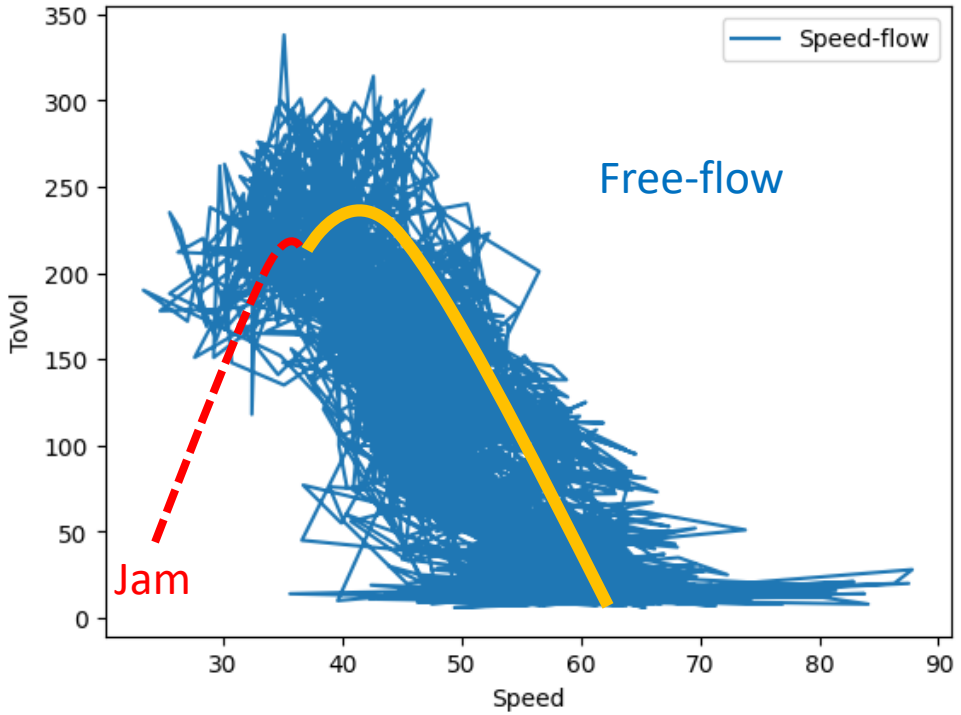
❖ 교통량이 많으면 속도 저하



- 교통량이 많아지면 속도 저하(자유류)
- 교통량이 용량을 초과하게 되면
  교통량 및 속도 감소(체증류)

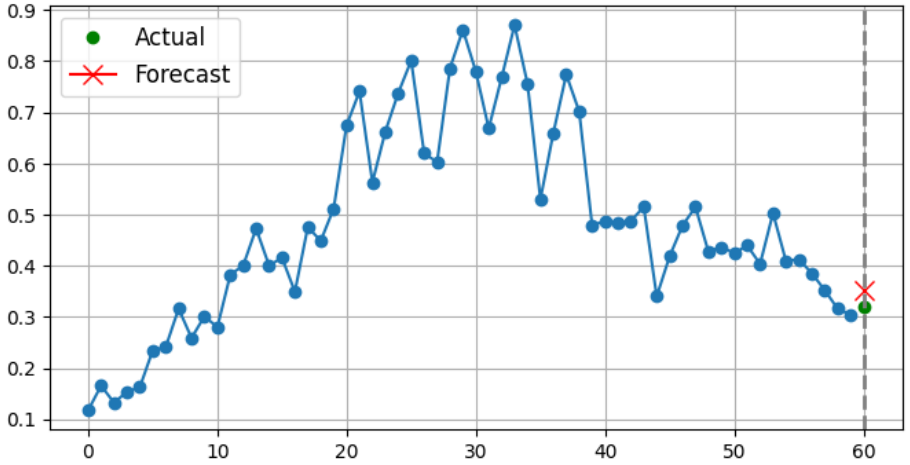
q_m : 최대교통류율(용량)
V_m : 임계속도

VDS16

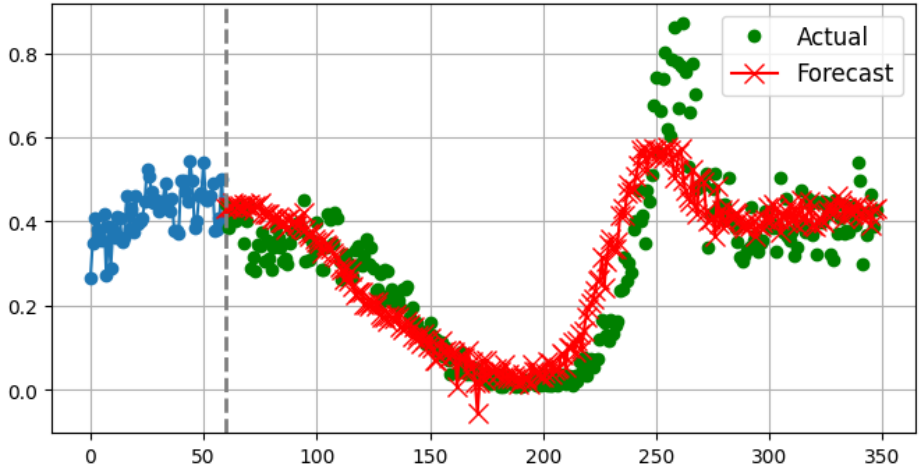


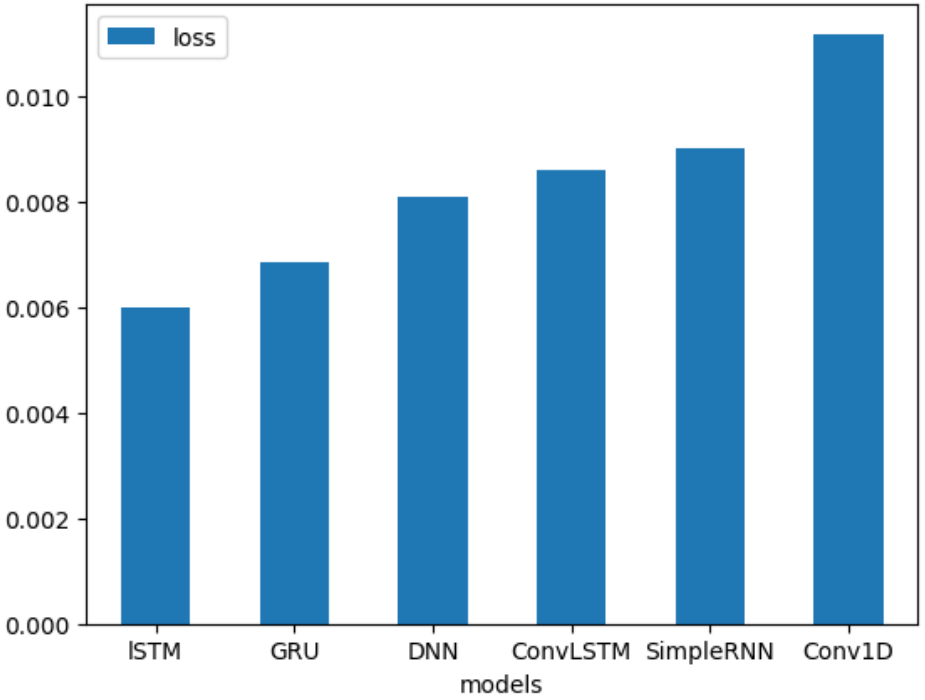
VDS 데이터 AI 모델 적용 예시 및 학습목표

LSTM 예측 : One-step ahead prediction



LSTM 예측 : Multi-step ahead prediction



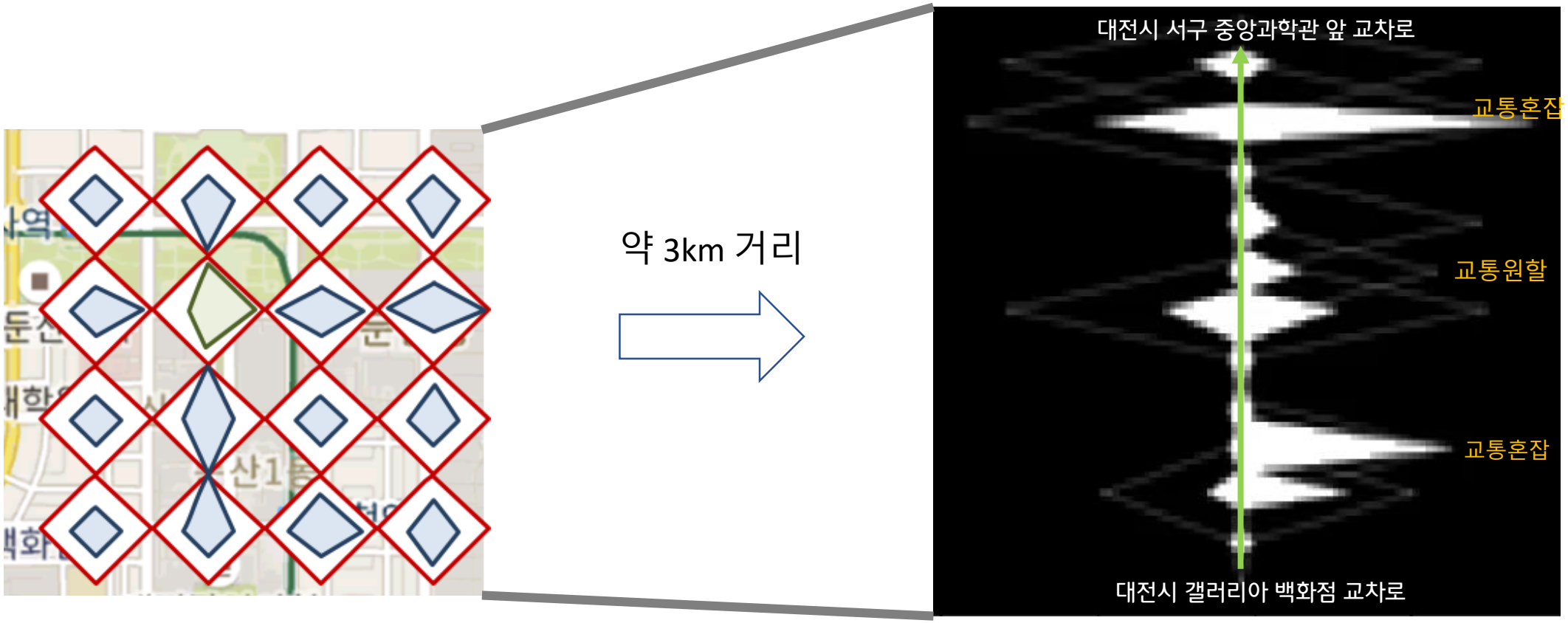


교통 데이터 활용한 AI 연구 소개 : CNN 적용

❖ RSE 교통 데이터를 직접 CNN을 적용하기 어렵다.

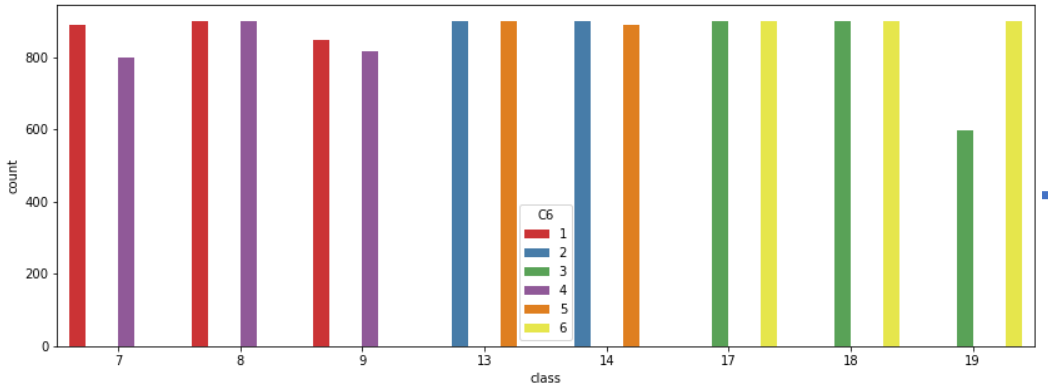
✓ RSE를 이미지로 변환하는 전략이 필요함

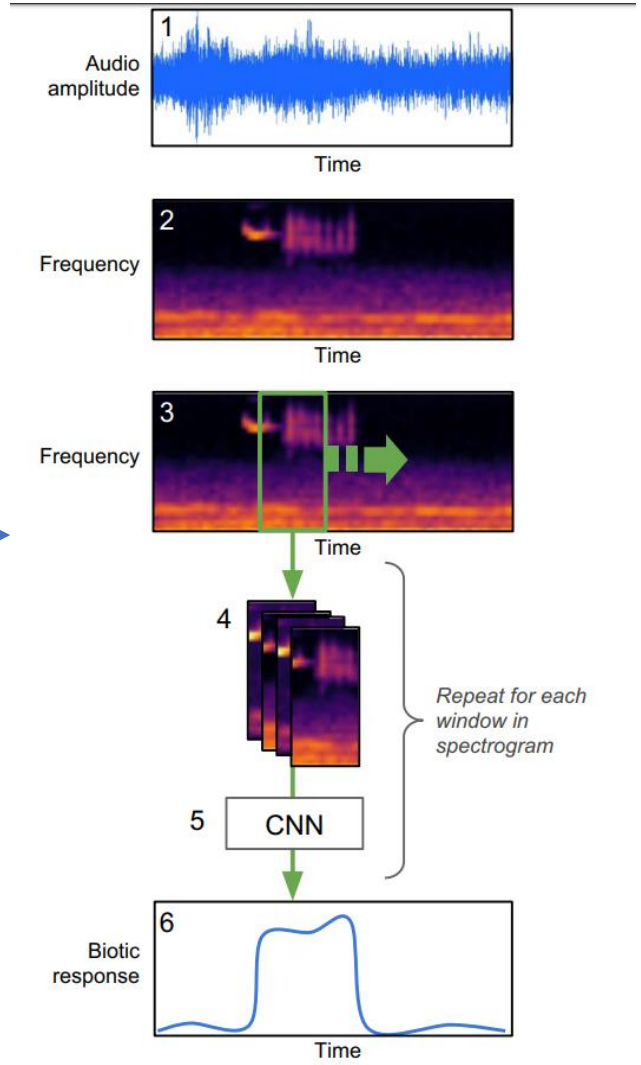
- 대전시 대덕대로 RSE 데이터



교통 데이터 활용한 AI 연구 소개 : CNN 적용

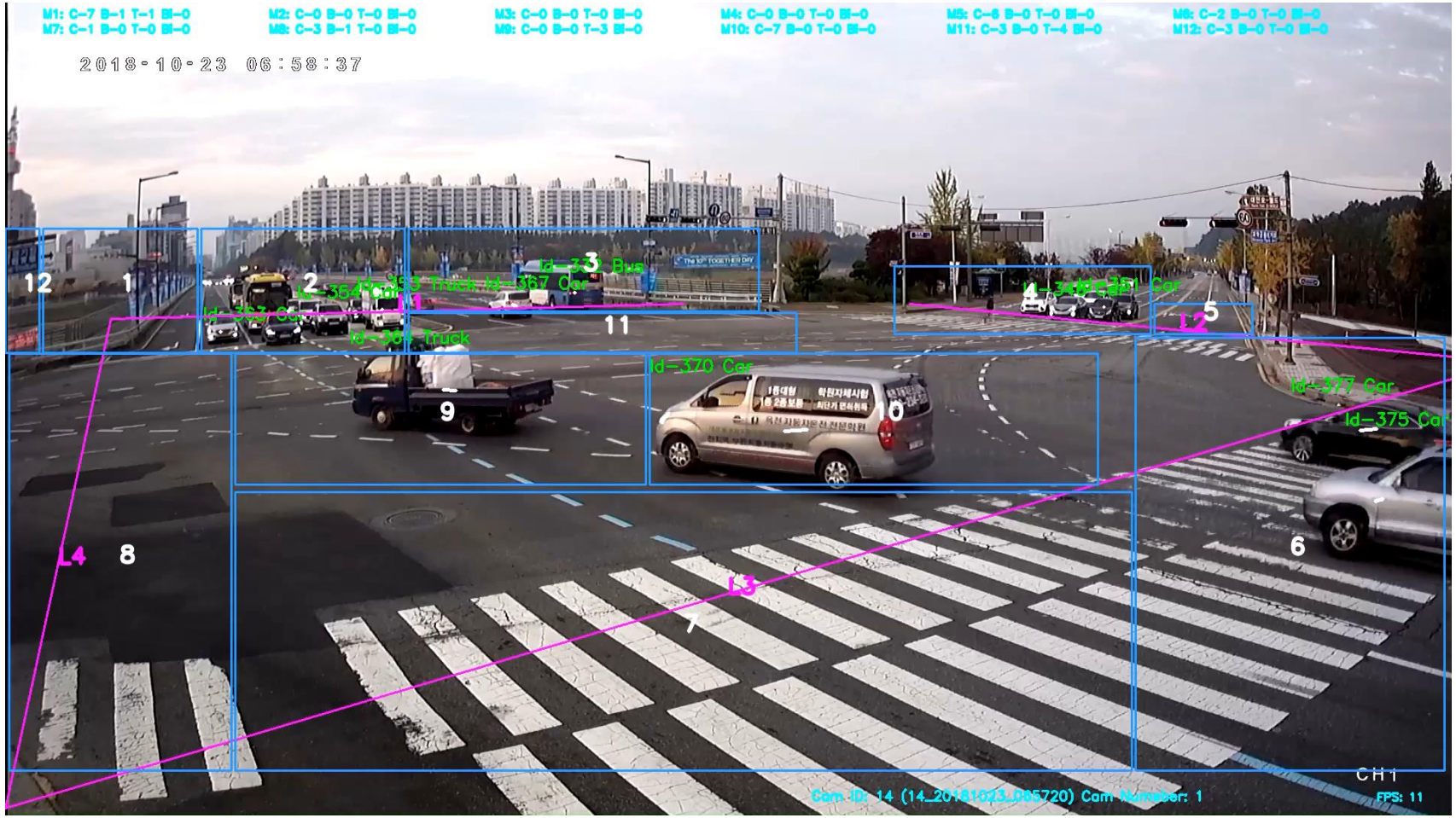
소음 빅데이터 자체 수집 : DJ_TrafficSound14K (대덕대로)





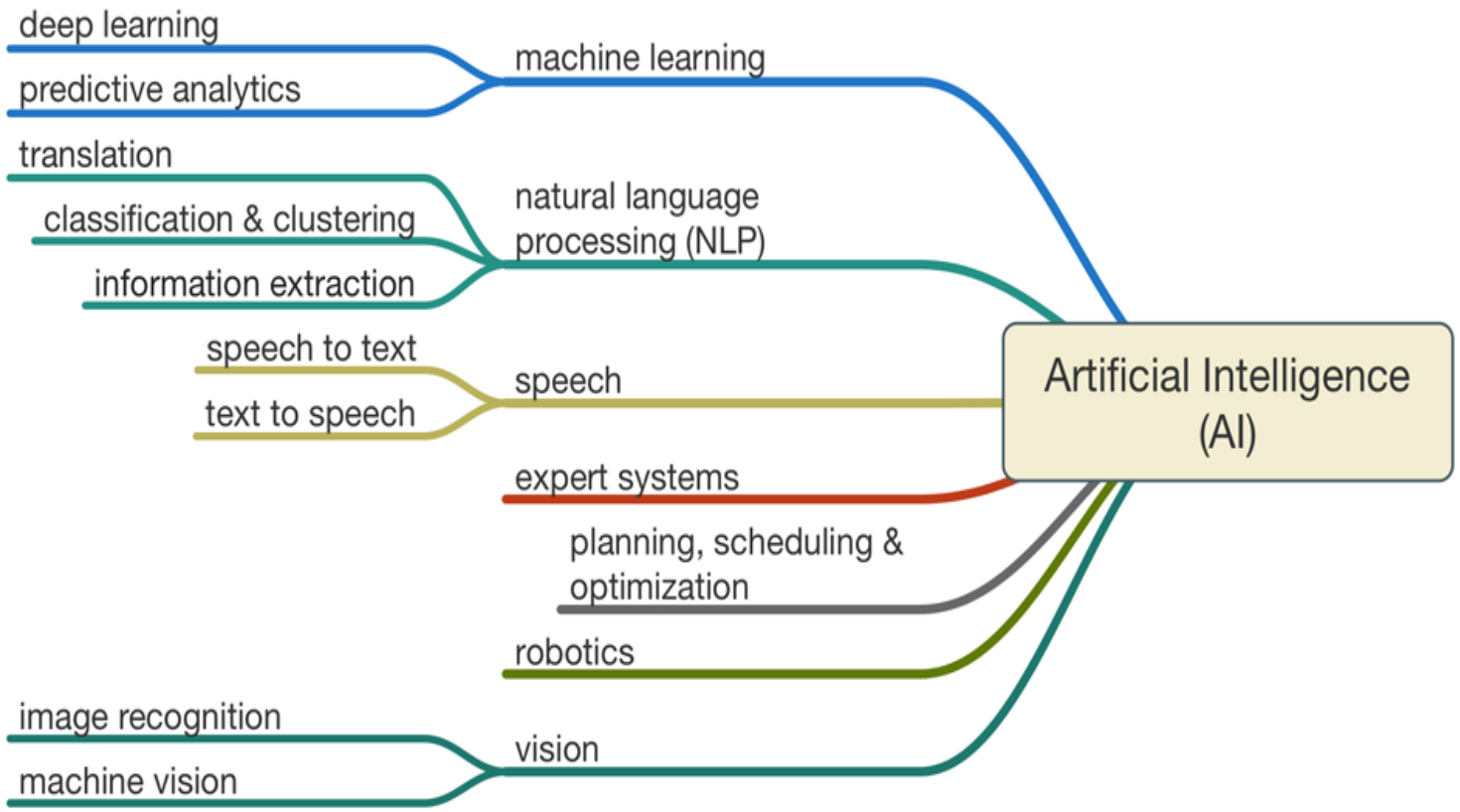
YOLO 적용: CCTV 영상에서 차량 탐색, 카운트, 추적 기술

- 세계 최고 AI 컨퍼런스인 CVPR20의 'AI City Challenge' 경진대회 참가 및 (7위, KISTI)



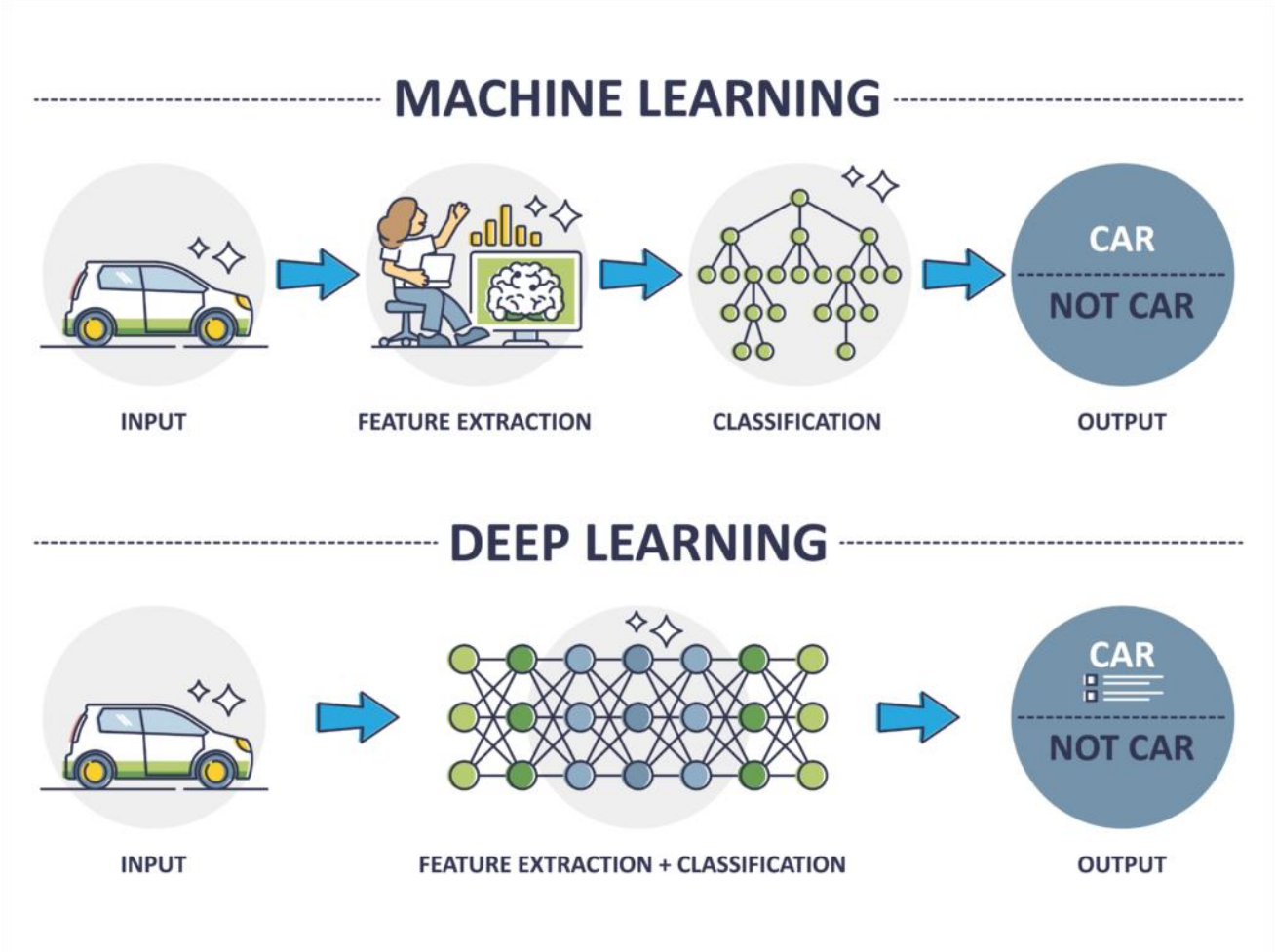
3) 머신러닝을 이용한 분류

인공지능의 분류



자료: https://www.eyerys.com/articles/paving-roads-artificial-intelligence-its-either-us-or-them

딥러닝과 머신러닝



(source) https://www.ait.de/en/deep-learning/

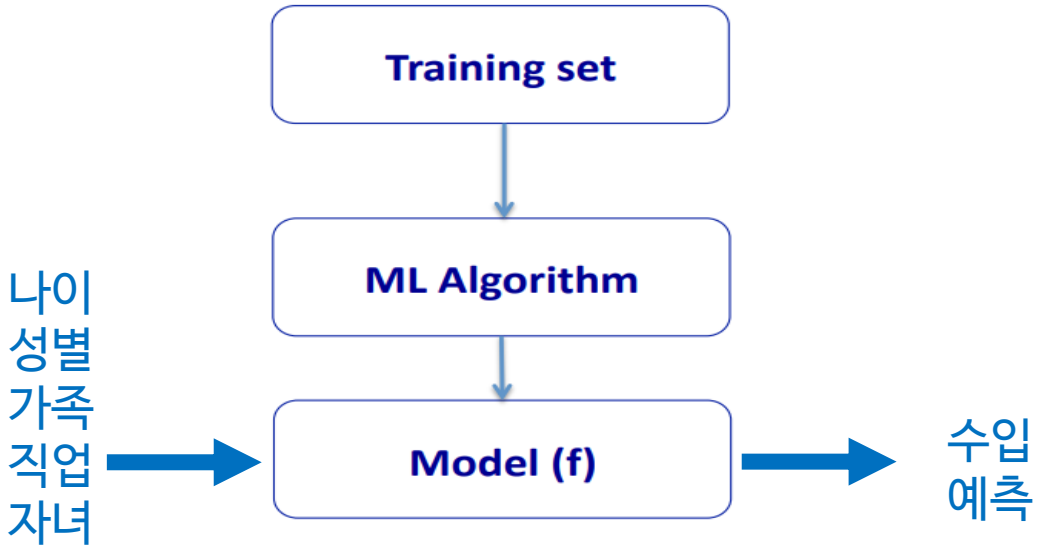
머신러닝 모델 프로세스

1) 데이터의 전처리 과정
   - ✓ 피처와 라벨을 결정함,
   - ✓ 필요한 피처를 추출하고 가공함,
   - ✓ 입력과 출력을 위한 피처의 정규화(표준화)

2) 머신러닝 모델을 선정
   - ✓ 학습, 검증, 테스트 데이터로 나누자
   - ✓ 적용할 머신러닝 알고리즘을 선정하자.
     - 분류 혹은 회귀 문제인지 먼저 정하자.

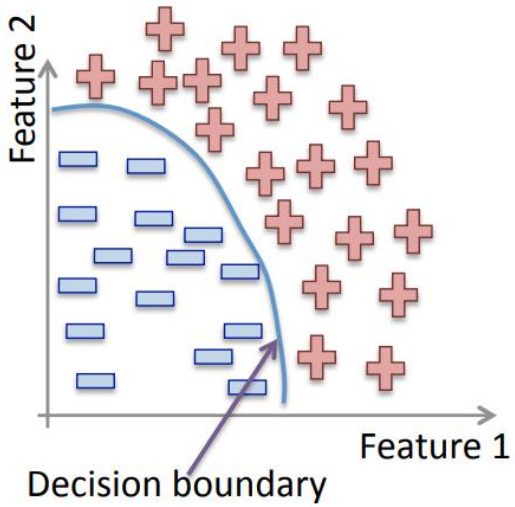
3) 모델 훈련
   - ✓ 모델을 훈련시키자.
   - ✓ 과적합 발생등 확인하자.

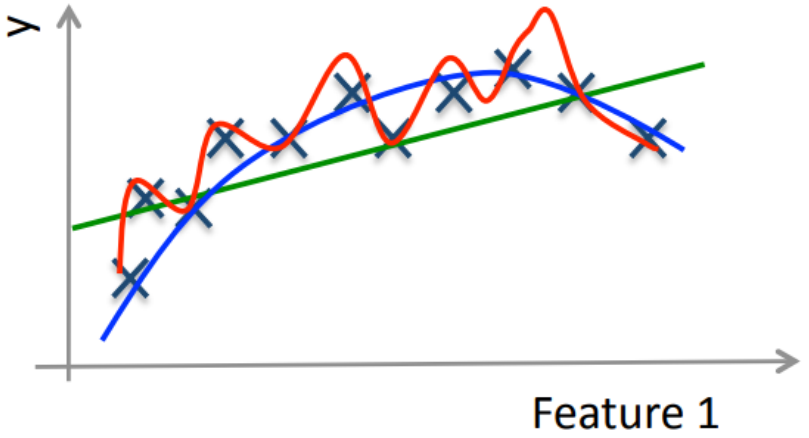
4) 모델 평가하자.
   - ✓ 모델의 테스트로 평가하자.



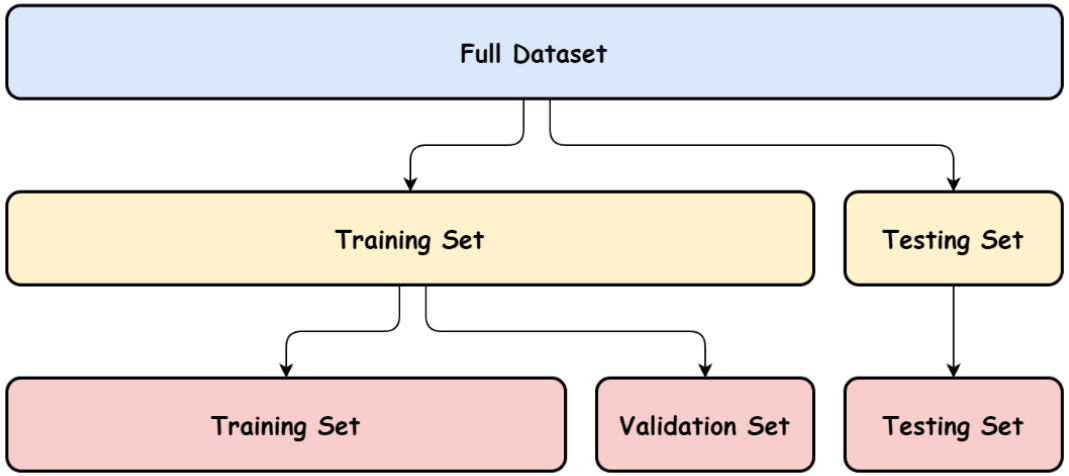
지도학습(Supervised Learning)

- 지도학습 알고리즘을 적용하기위해서는 레이블이 있어야 함
  - ✓ 누가 레이블을 만드나?
- 분류
  - ✓ 분류의 정확도
- 회귀
  - ✓ 실측치와 출력치의 오차를 최소화





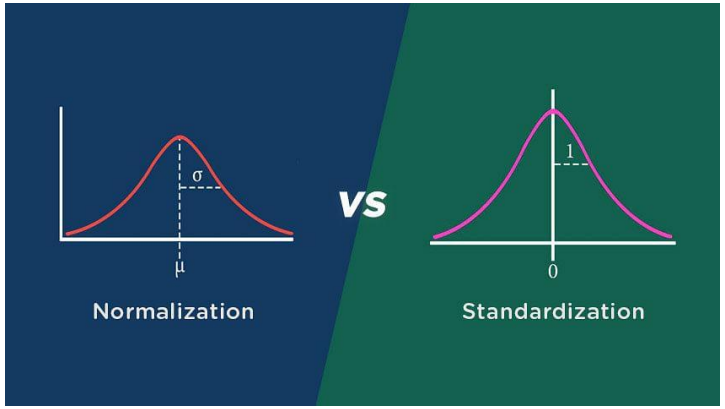
Splitting Your Data: Training, Testing, and Validation Datasets



학습을 잘하기 위해서 입력 데이터 처리 : 피처 스케일링과 정규화

- 피처 스케일링(feature scaling): 서로 다른 변수의 값 범위를 일정하게 맞추는 작업
  - ✓ 표준화(Standardization): 데이터 피처 각각이 평균이 0이고 분산이 1인 가우시안 분포로 변환
  - ✓ 정규화(Normalization) : 서로 다른 피처의 크기를 통일하기 위해 크기를 변환 (0~1 사이)

$$x_i\_new = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

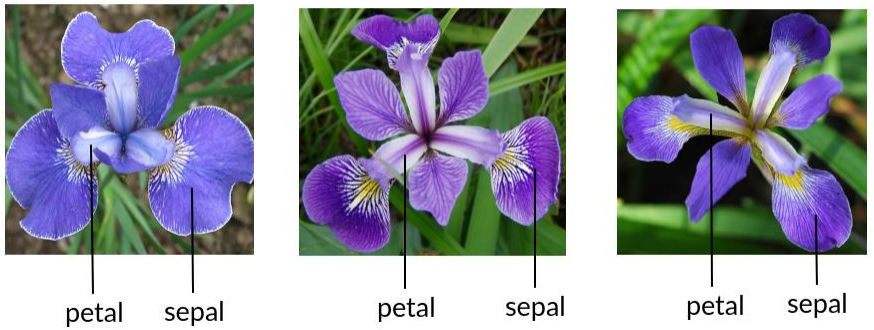


$$x_i\_new = \frac{x_i - mean(x)}{stdev(x)}$$

(그림출처) https://www.simplilearn.com/normalization-vs-standardization-article

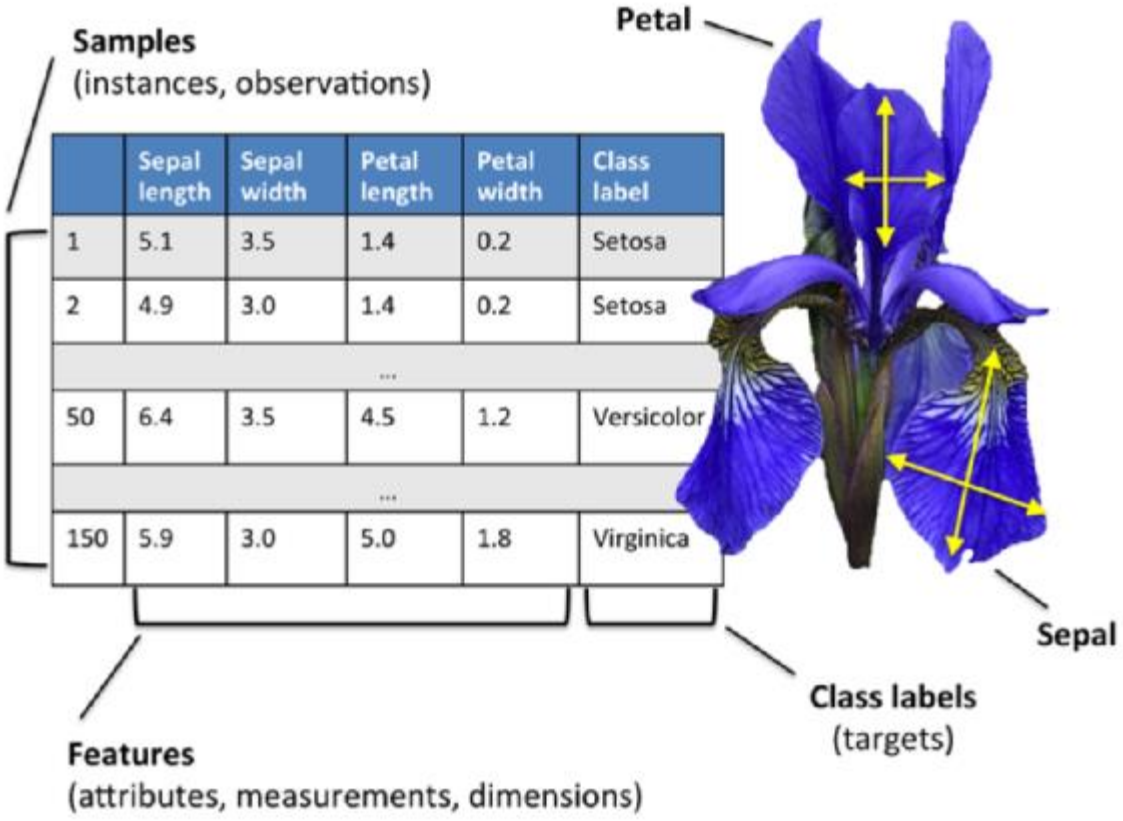
초보자가 인공지능 전문가 되기 위한 방법: 모방하자

Kaggle에서 잘 된 코드를 기본으로, 데이터 구조를 유사하게 만들어 사용하자.



Train/test split : 20%

X_train: (120, 4, 1)
y_train: (120,)
X_test : (30, 4, 1)
y_test : (30,)

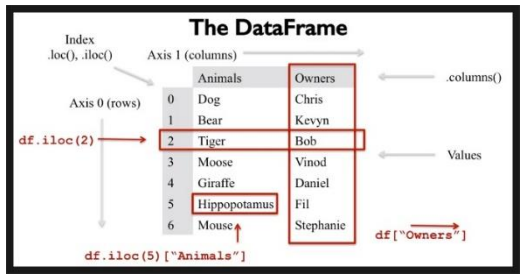


source : https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_features_extraction.php

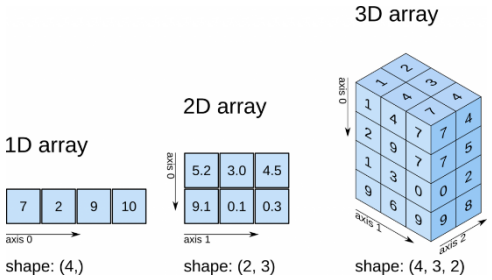
머신러닝에 많이 사용되는 도구(라이브러리)

pandas

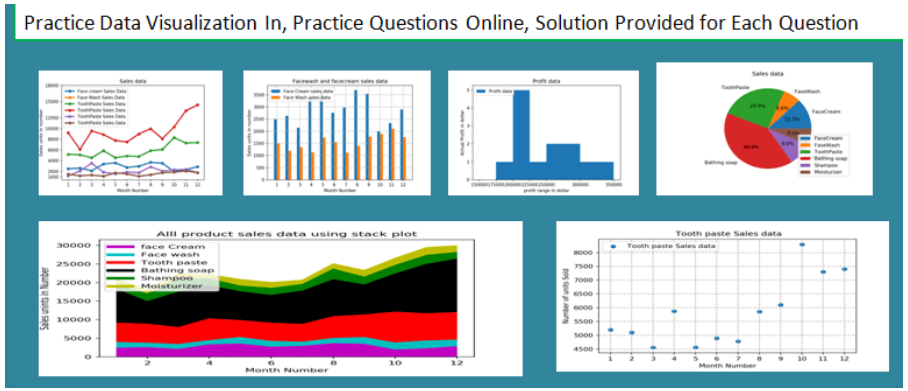
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



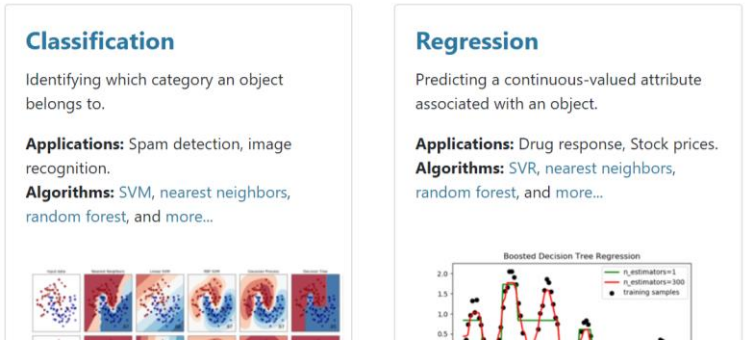
NumPy



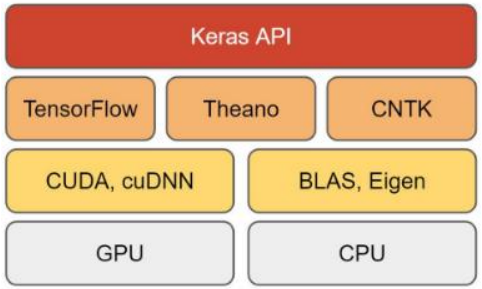
Python Matplotlib



scikit-learn



TensorFlow PYTORCH



jupyter Python Notebook

Pandas를 이용하여 VDS 데이터 불러오기

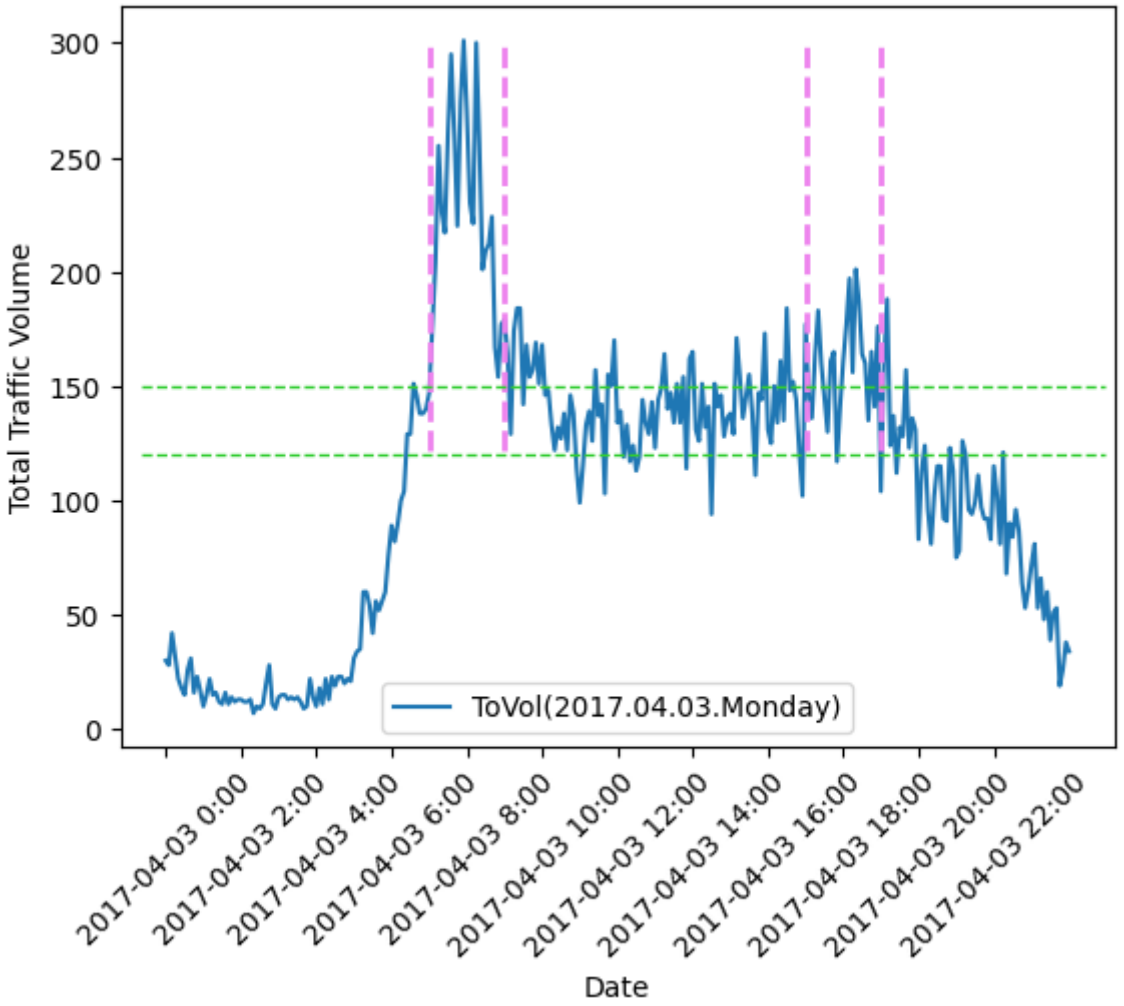
```
import pandas as pd
```

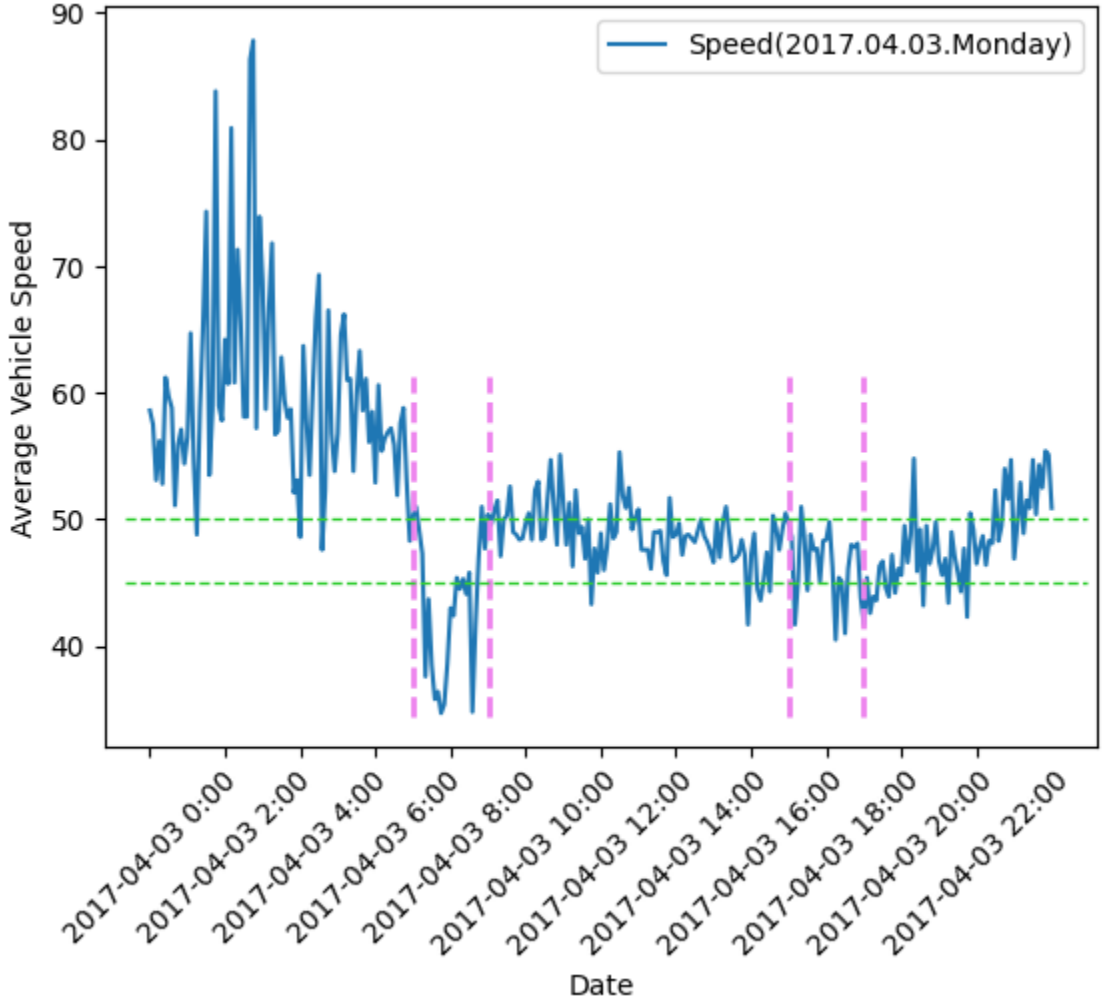
```
df = pd.read_csv("https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/vds16.csv")
```

```
df.head()
```

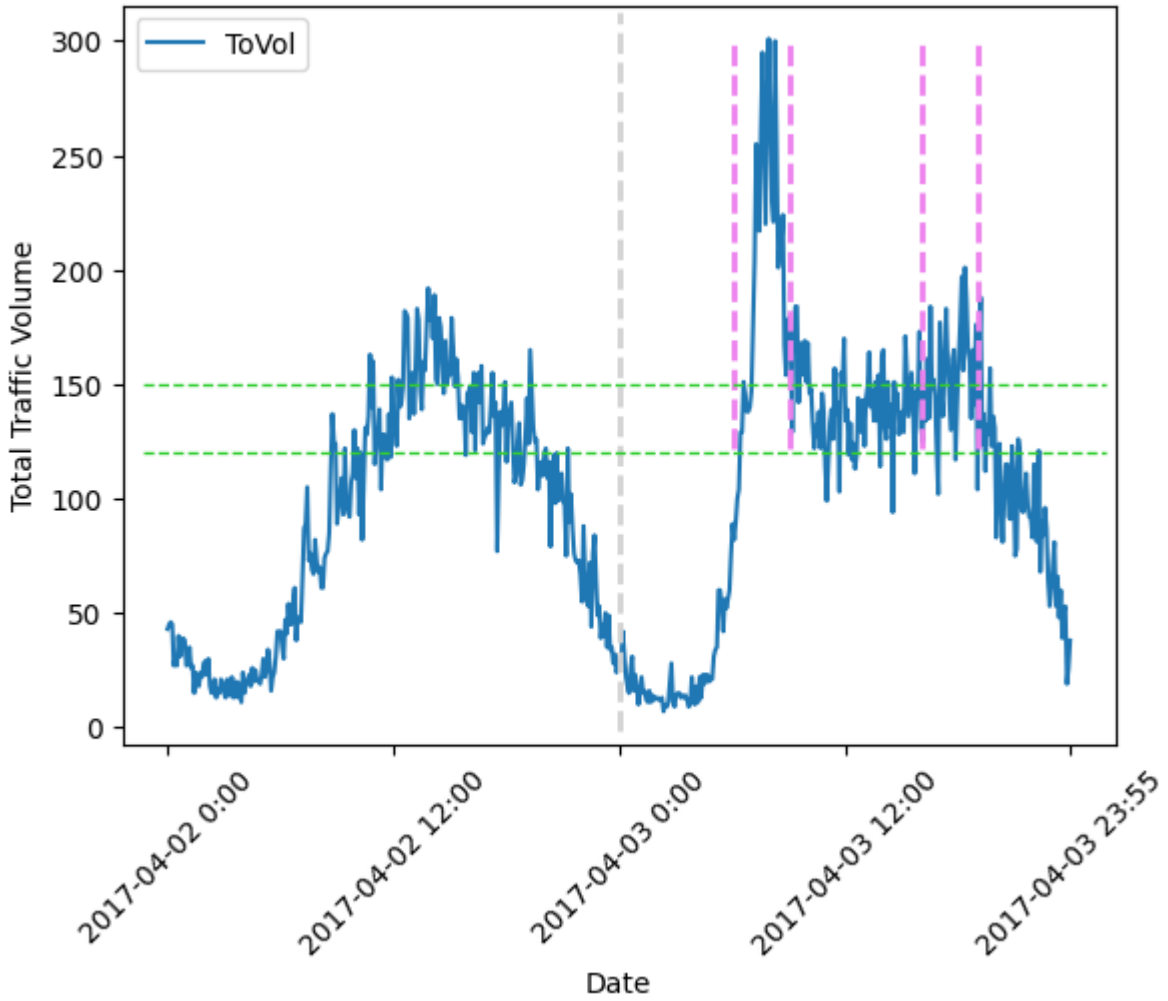
	Date	ToVol	SmVol	MeVol	LaVol	Speed	OccRate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84
2	2017-04-02 0:10	46	34	12	0	50.6	1.87
3	2017-04-02 0:15	45	36	9	0	50.9	1.72
4	2017-04-02 0:20	27	13	13	1	62.2	1.12

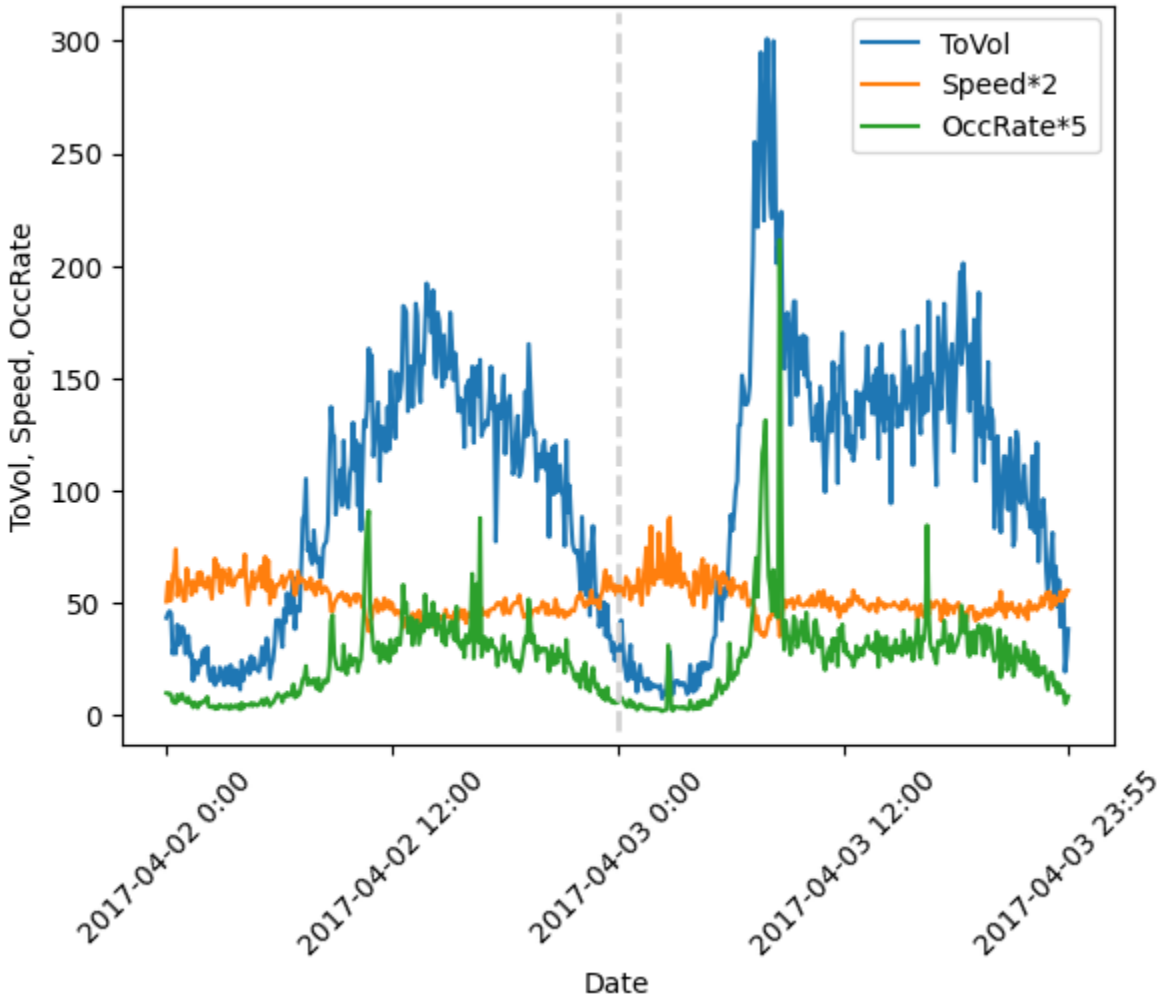
Matplotlib을 이용하여 특성 가시화





Matplotlib을 이용하여 특성 가시화





VDS 데이터 라벨링 전략

KISTI SINCE 1962

❖ 교통흐름을 라벨링 해보자

- ✓ 라벨을 몇 개로 할 것인가?
- ✓ 무엇으로 라벨을 할 것인가?
  - 속도, 교통량, 점유률 등

```
num_classes = 3
class_labels= ['Jam', 'Slow', 'Normal']
```

```
def get_score(speed):
    if speed < 45:
        label = 'Jam'
    elif speed < 50:
        label = 'Slow'
    else :
        label = 'Normal'
    return label
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

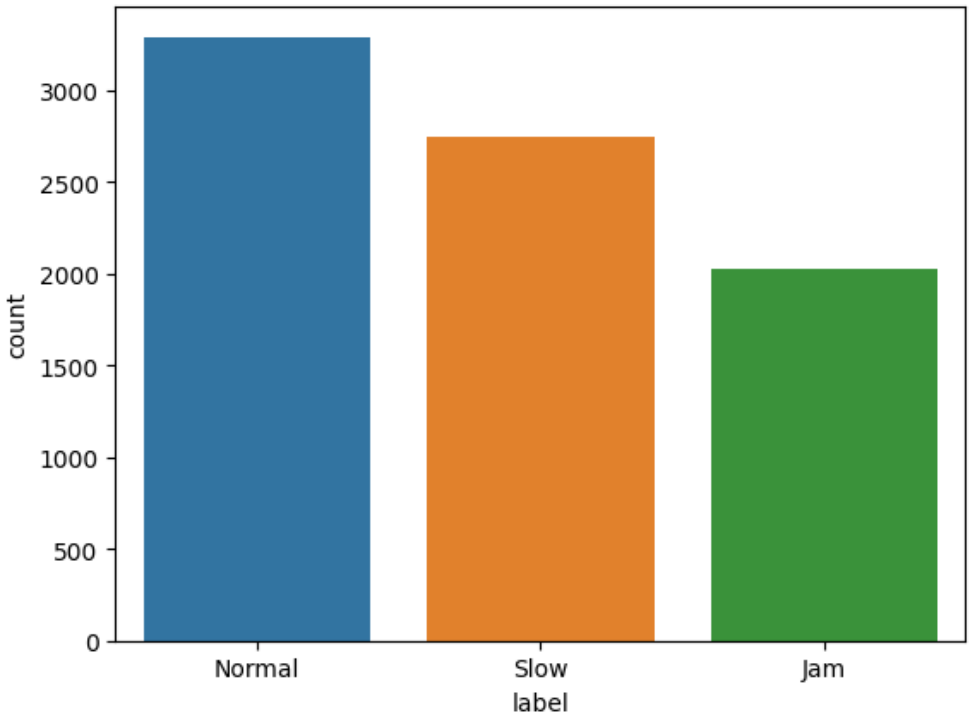
	Date	ToVol	SmVol	MeVol	LaVol	Speed	OccRate	label
0	2017-04-02 0:00	43	34	9	0	50.3	1.90	Normal
1	2017-04-02 0:05	45	32	13	0	58.9	1.84	Normal
2	2017-04-02 0:10	46	34	12	0	50.6	1.87	Normal
3	2017-04-02 0:15	45	36	9	0	50.9	1.72	Normal
4	2017-04-02 0:20	27	13	13	1	62.2	1.12	Normal

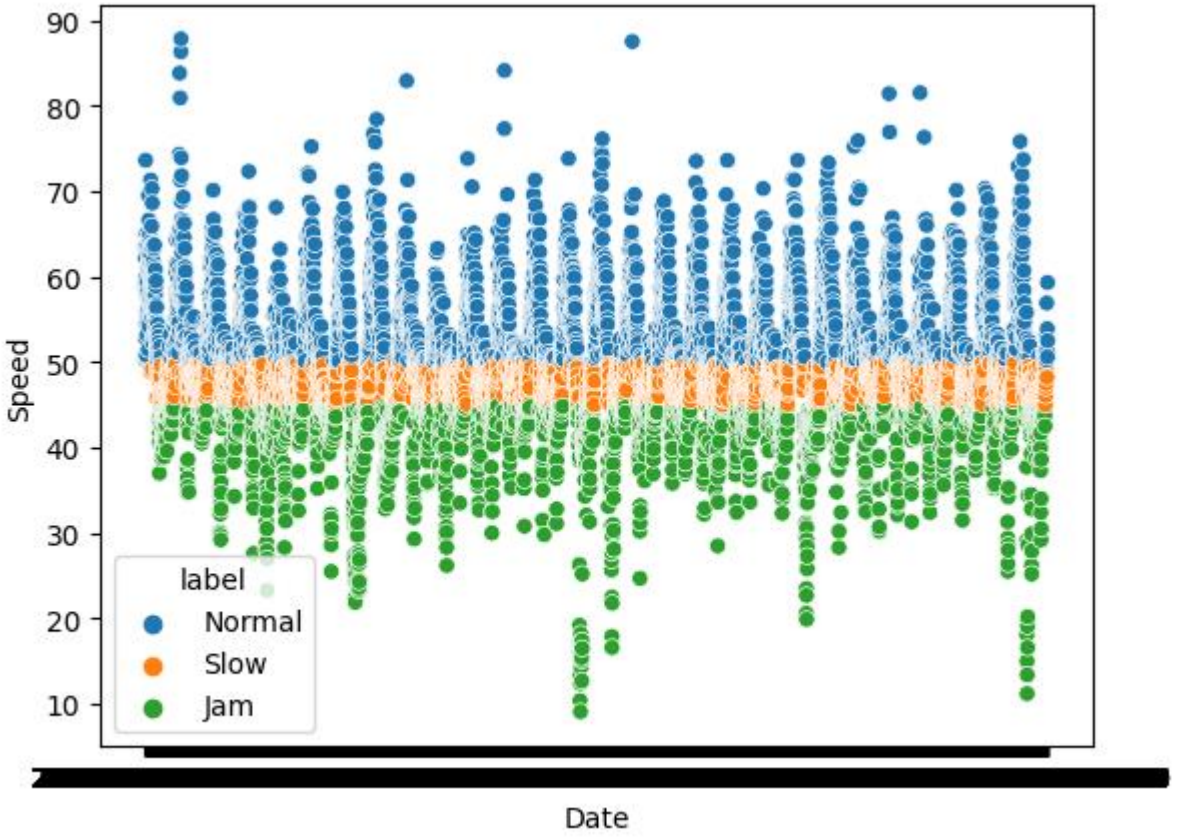
```
label = df['label'].unique()
label
```

```
array(['Normal', 'Slow', 'Jam'], dtype=object)
```

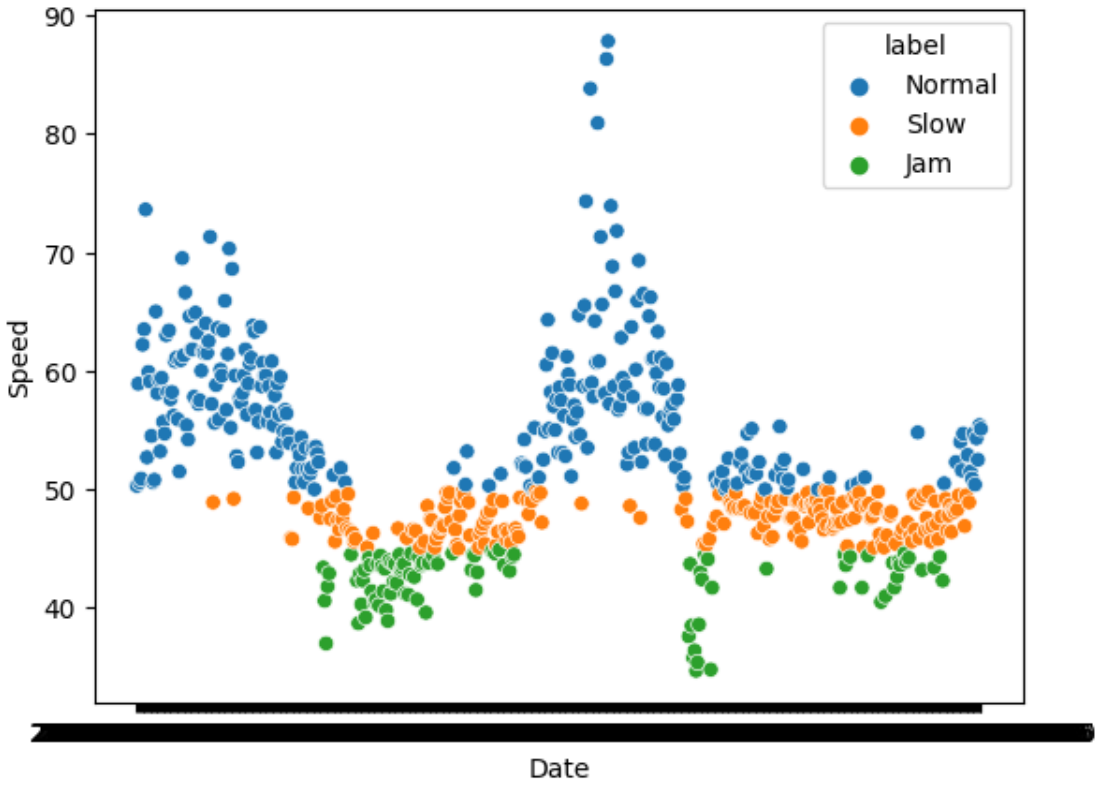
Seaborn을 이용한 상관관계 등 가시화

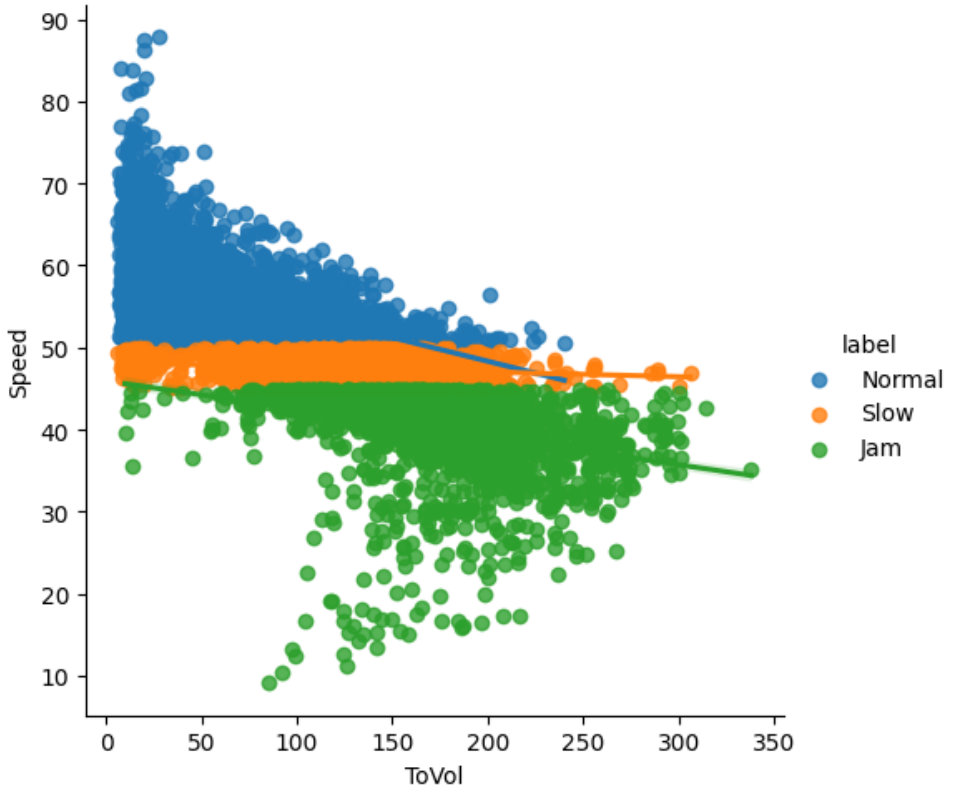
라벨이 잘 되었는지 논의해본다.



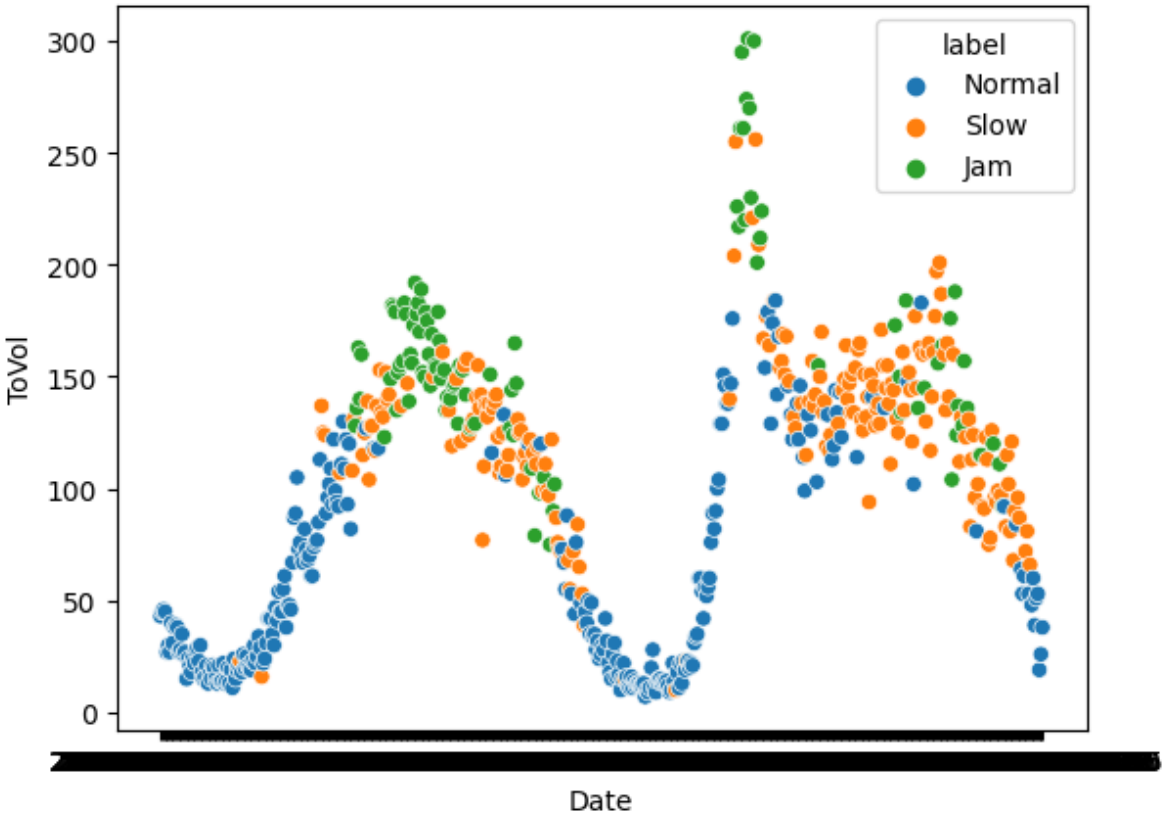


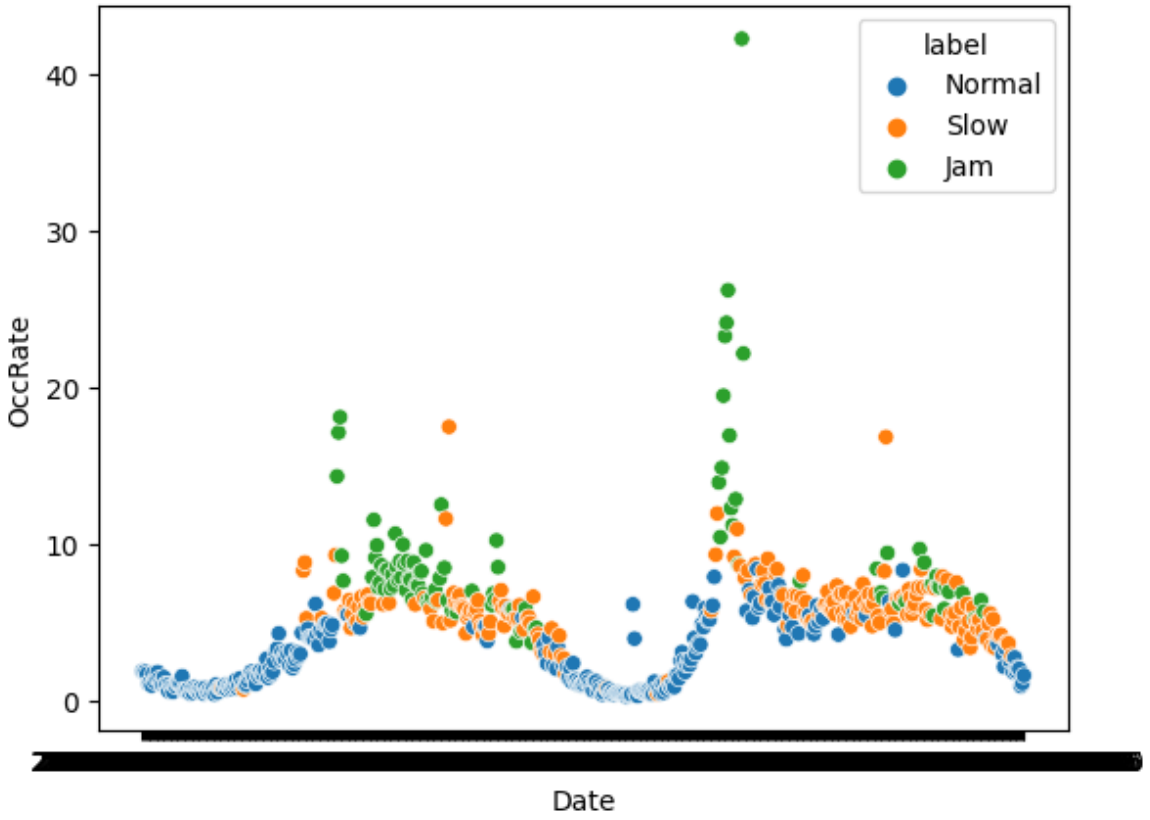
Seaborn을 이용한 상관관계 등 가시화



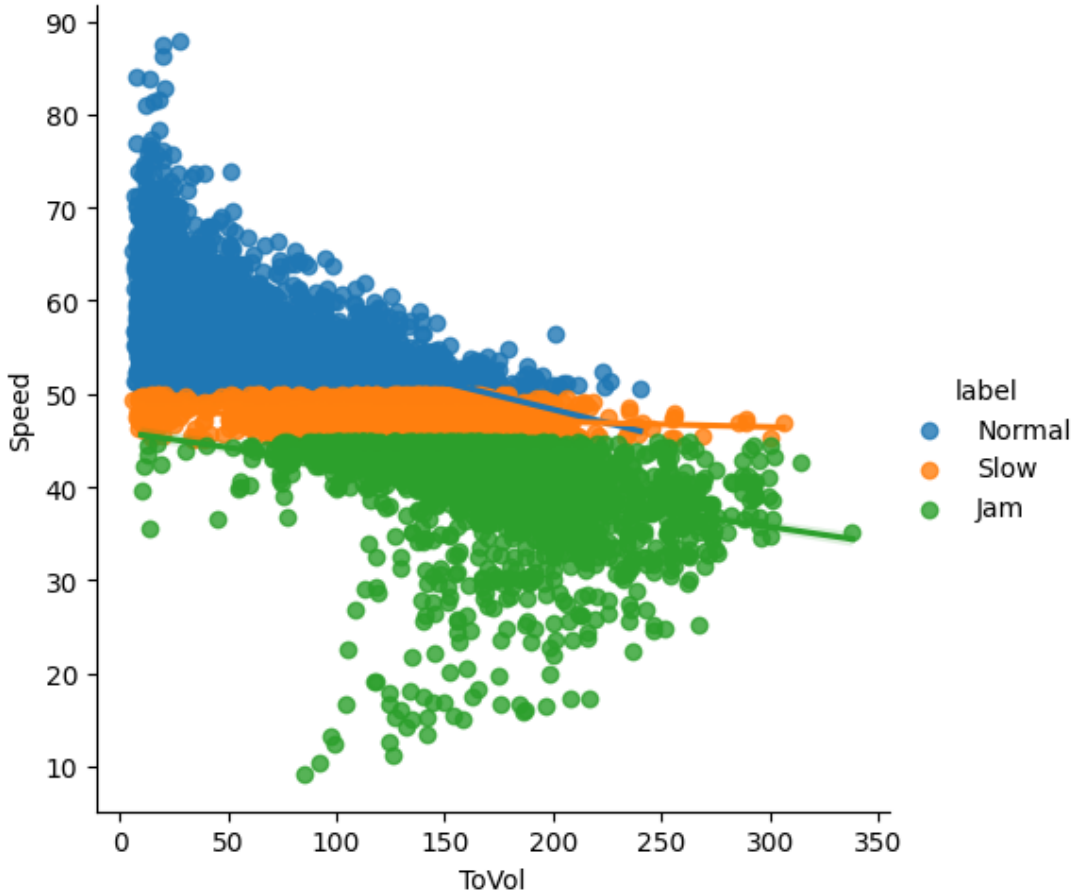


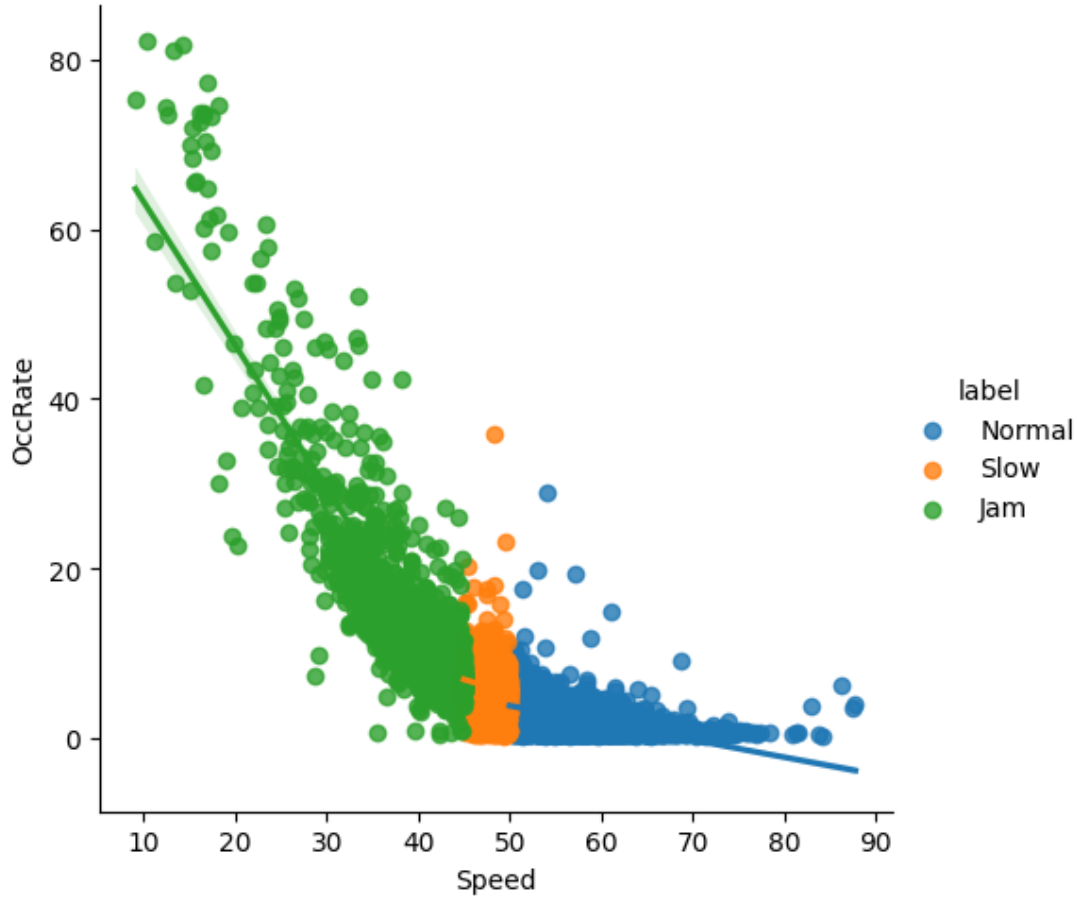
Seaborn을 이용한 상관관계 등 가시화



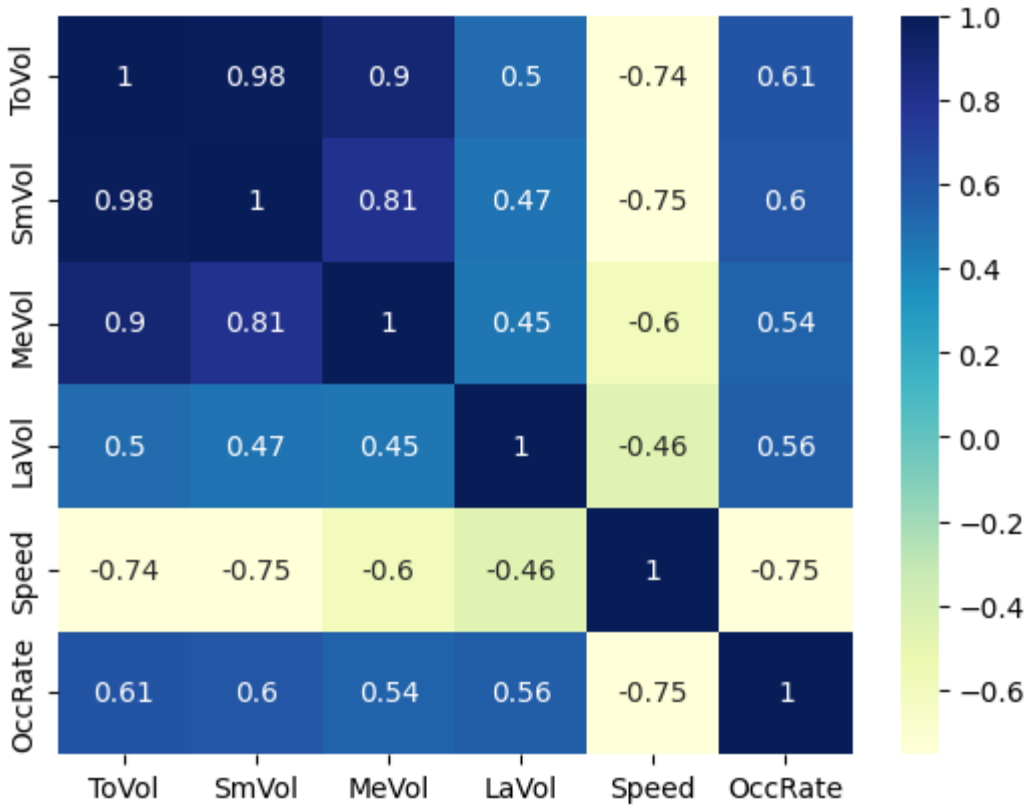


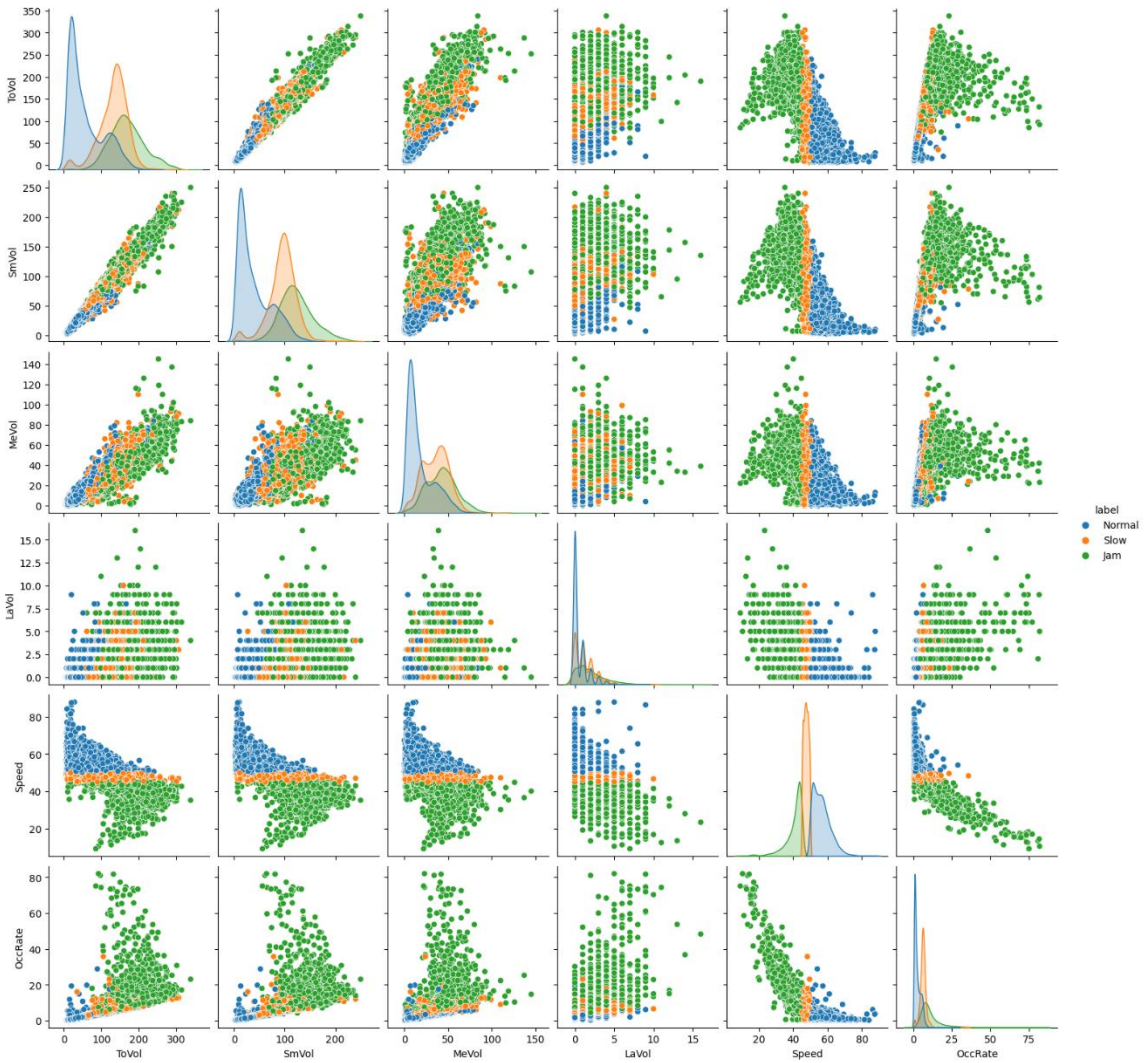
Seaborn을 이용한 상관관계 등 가시화





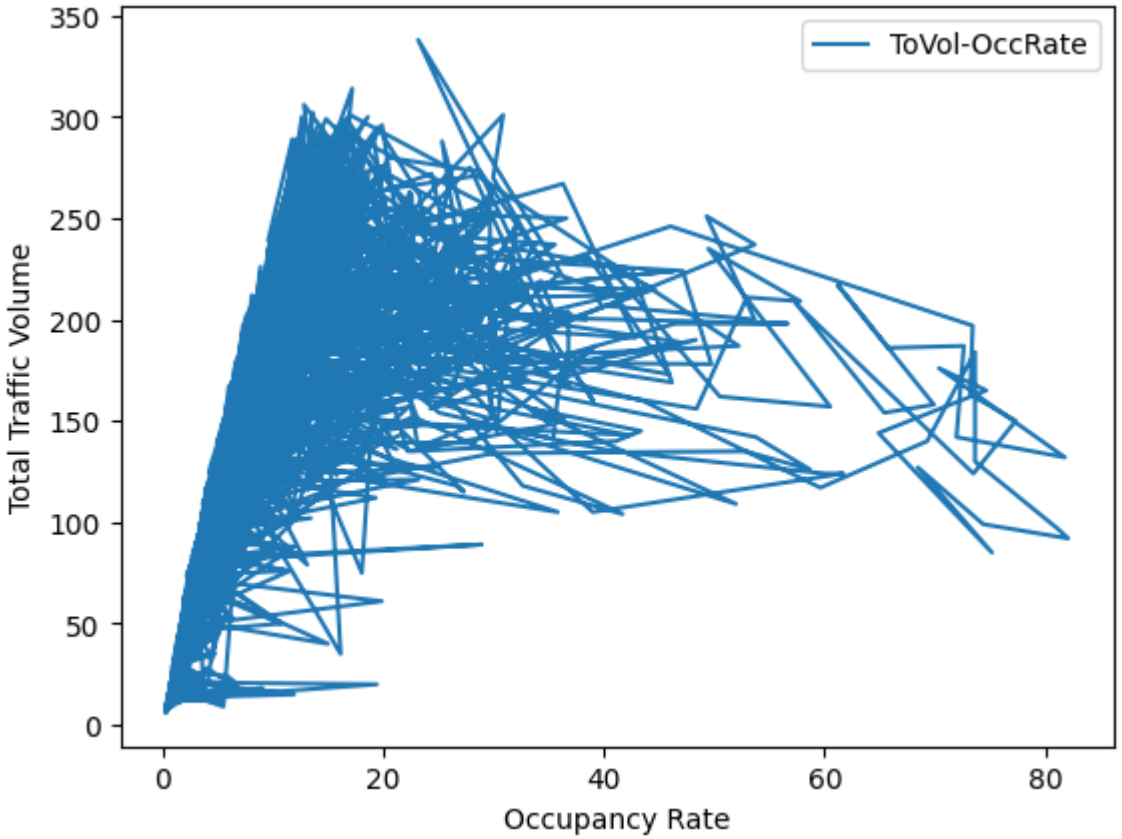
Seaborn을 이용한 상관관계 등 가시화



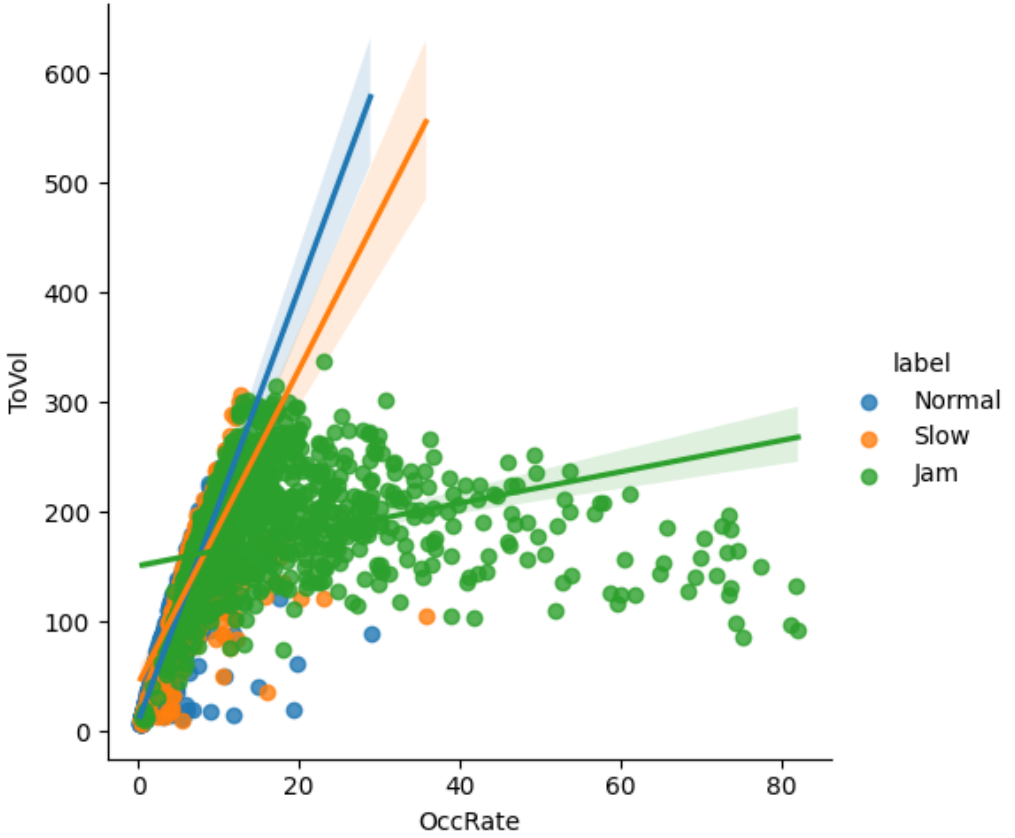


대전시 유성구 VDS16 도로 교통량-점유률 관계

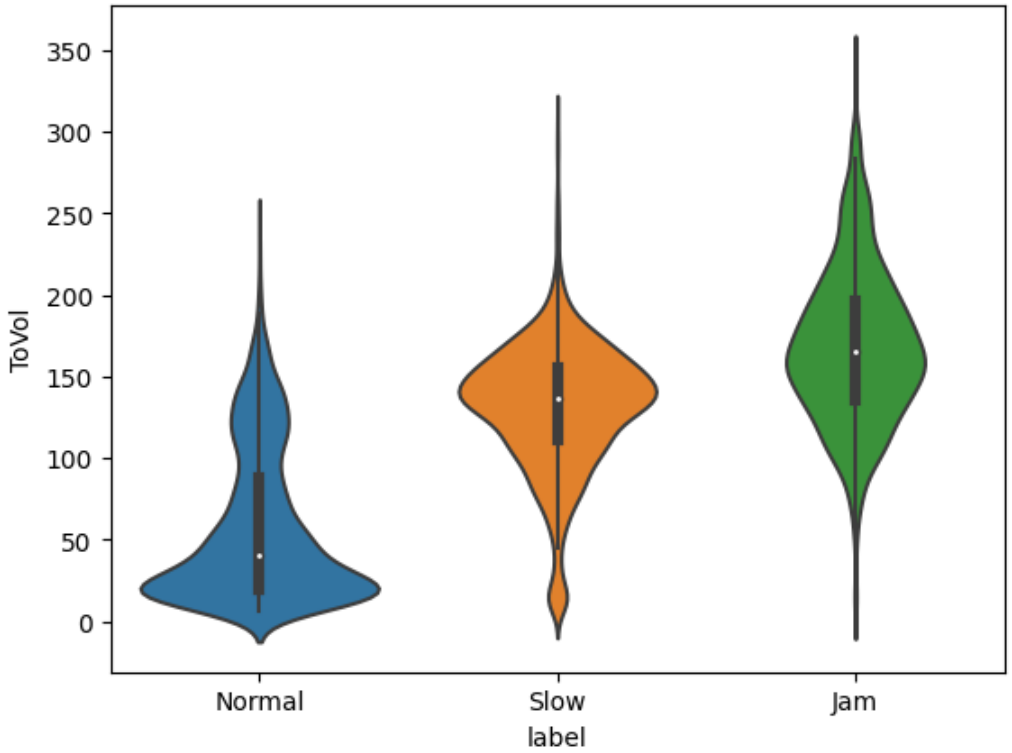
라벨링 이전

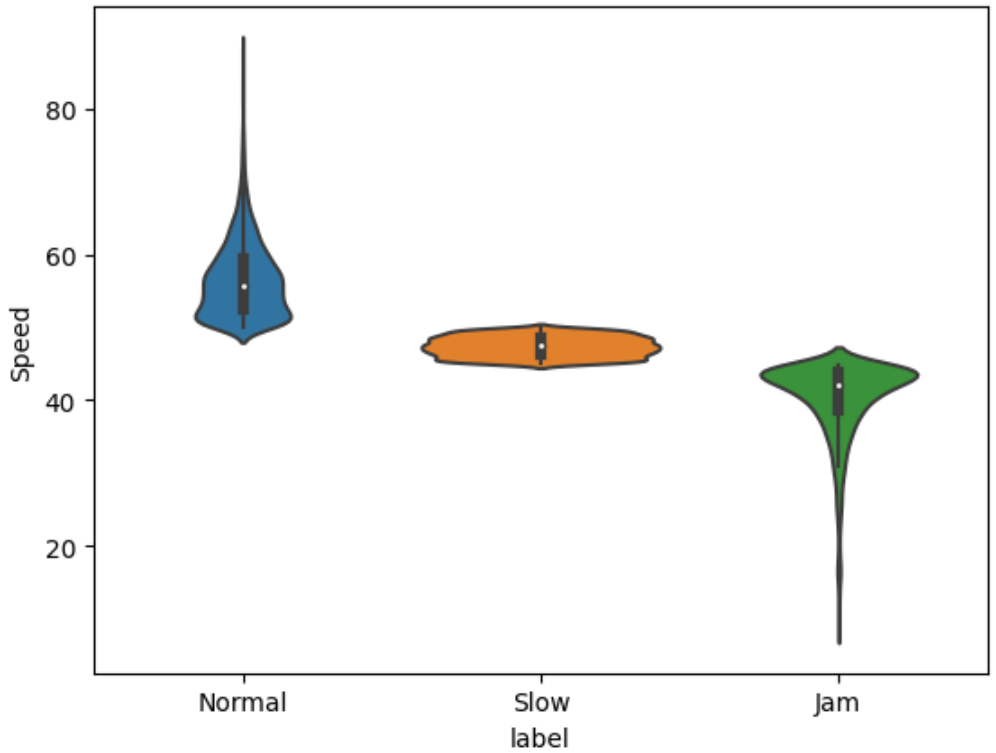


라벨링 이후



Seaborn을 이용한 상관관계 등 가시화: violinplot



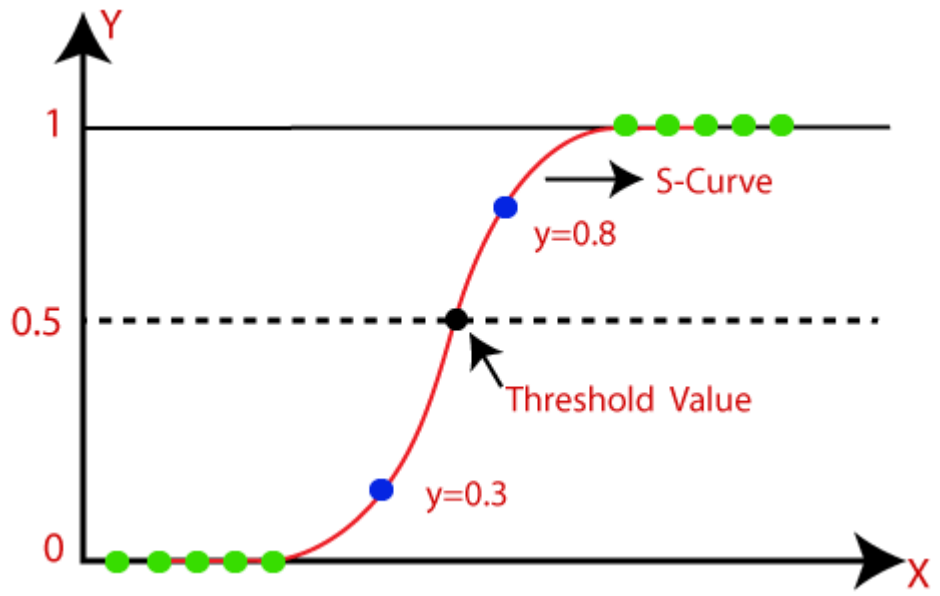


4) 머신러닝을 적용해보자

Logistic regression, Decision tree, Random forest, KNN, SVM, GBoost

Logistic regression은 분류 모델로 사용가능

- Logistic regression은 라벨이 범주형일때 사용한다.
  - ✓ 시그모이드 함수와 같다.



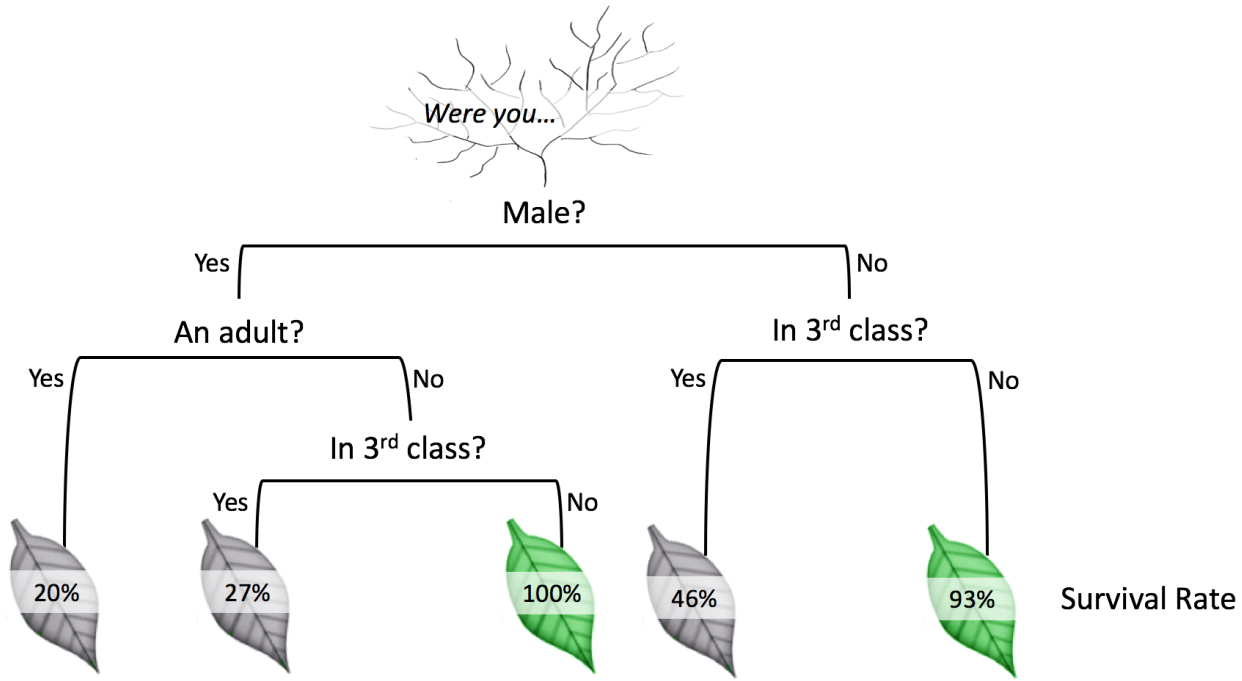

```
# LogisticRegression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       600
           1       1.00      0.99      0.99       979
           2       0.98      1.00      0.99       841

    accuracy                           0.99      2420
   macro avg       0.99      0.99      0.99      2420
weighted avg       0.99      0.99      0.99      2420

[[594   0   6]
 [  0 966  13]
 [  1   3 837]]
accuracy is 0.990495867768595
```

그림출처) http://localhost:8888/notebooks/lab_vds/vds01_machinelearning.ipynb

Decision Tree: 의사결정나무

- 의사결정나무는 의사결정을 도와주는 툴로 나무 같은 그래프 모델을 사용한다.
  - ✓ 20고개 같은 방식
  - ✓ 각각의 잎 노드(leaf)는 클래스 라벨(범주)를 나타낸다.




```
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print('accuracy is',accuracy_score(y_pred,y_test))
```

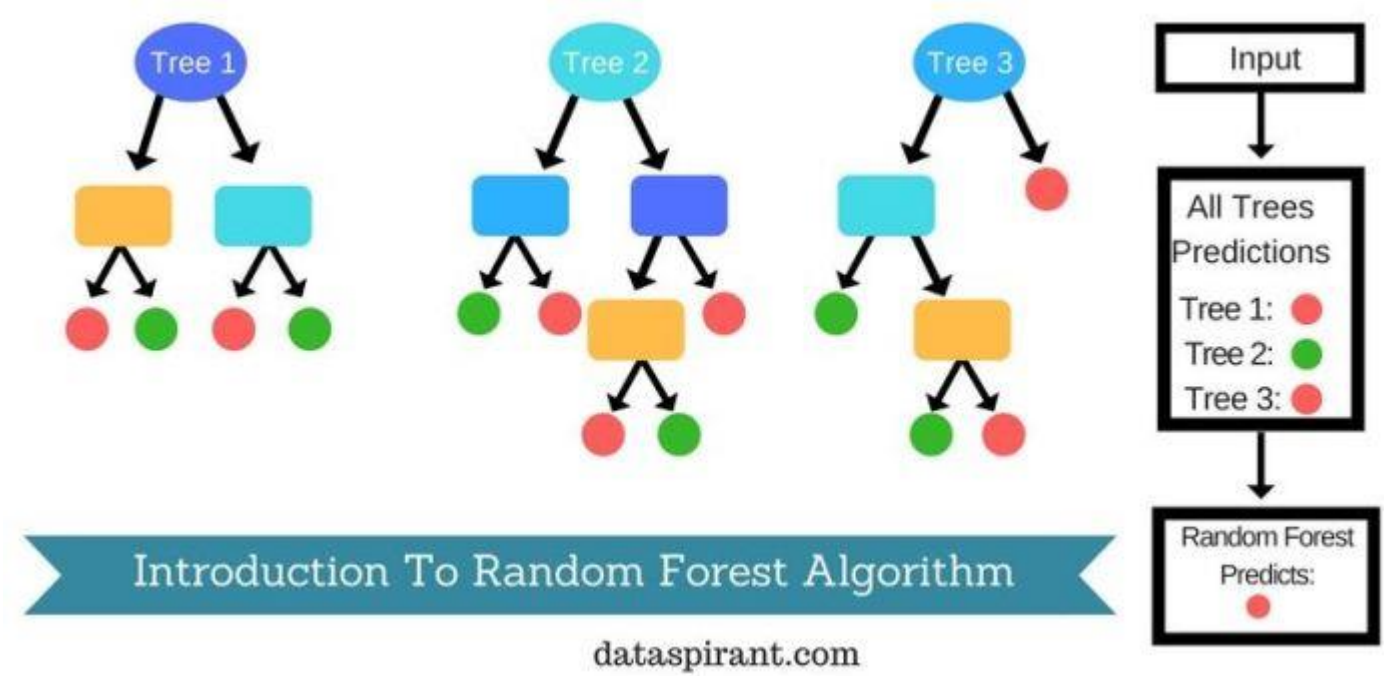
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       600
           1       1.00      1.00      1.00       979
           2       1.00      1.00      1.00       841

    accuracy                           1.00      2420
   macro avg       1.00      1.00      1.00      2420
weighted avg       1.00      1.00      1.00      2420

[[600   0   0]
 [  0 979   0]
 [  0   0 841]]
accuracy is 1.0
```

그림출처) http://localhost:8888/notebooks/lab_vds/vds01_label_visualization.ipynb

Random Forest 분류

❖ 램덤포래스트는 의사결정나문의 집합으로

✓ 의사결정 나무들 투표로 테스트 라벨의 최종 범주를 결정한다.



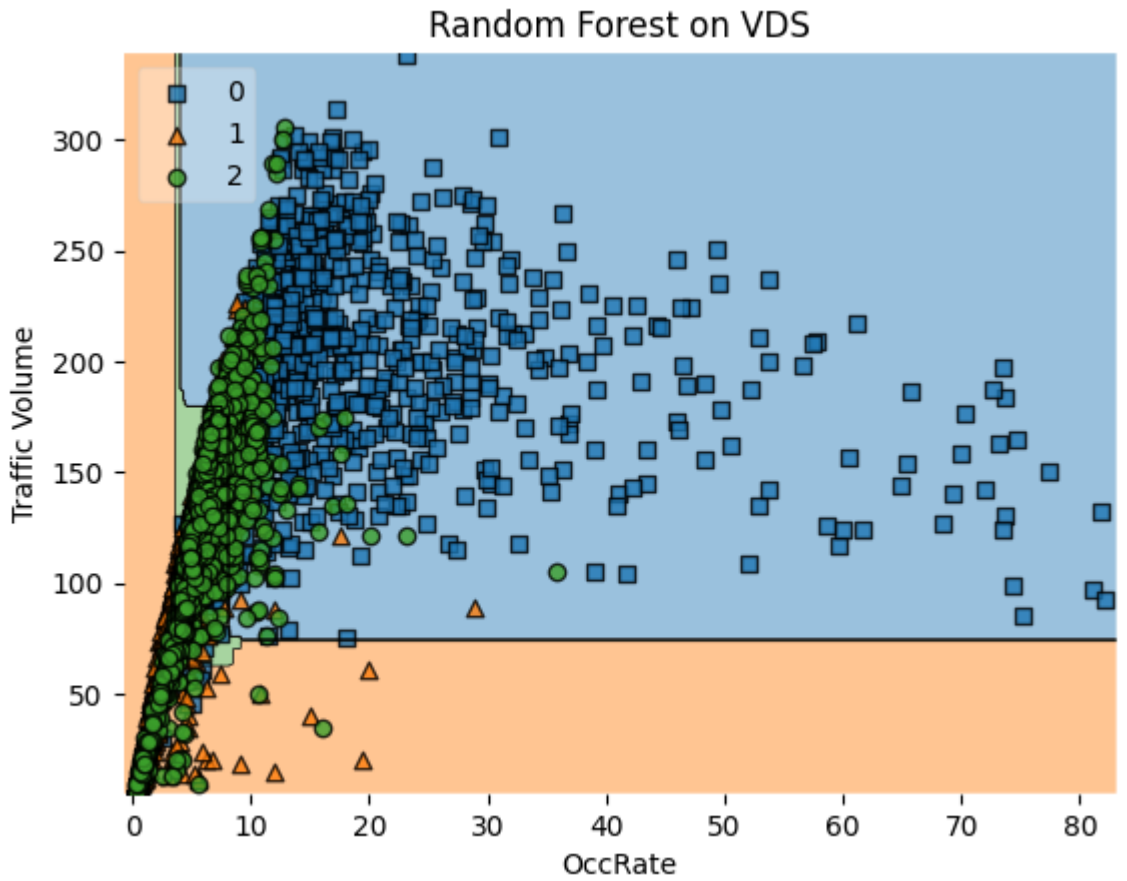
그림출처) http://localhost:8888/notebooks/lab_vds/vds01_label_visualization.ipynb

Random Forest 분류

```
Model=RandomForestClassifier(max_depth=2)
Model.fit(X_train,y_train)
y_pred=Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_pred,y_test))
#Accuracy Score
print('accuracy is ',accuracy_score(y_pred,y_test))
```

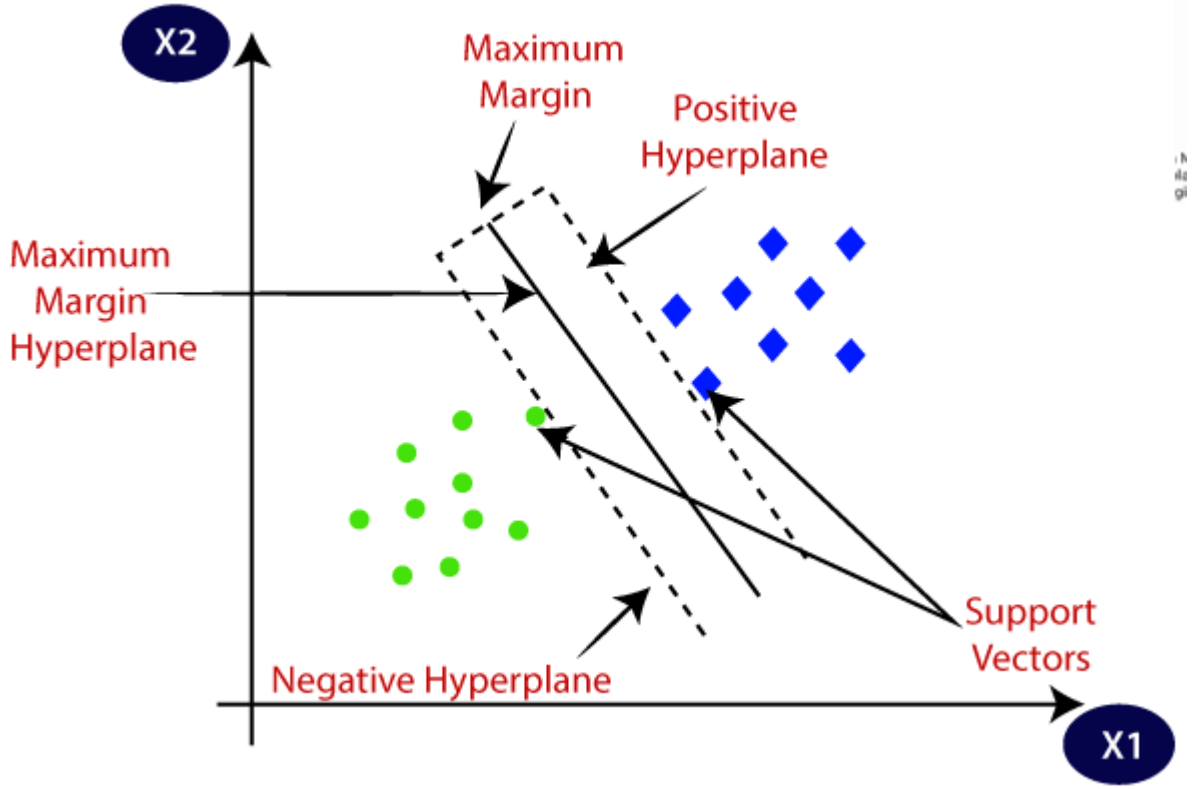
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       600
           1       1.00      0.81      0.89       979
           2       0.81      0.99      0.89       841

    accuracy                           0.92      2420
   macro avg       0.93      0.93      0.92      2420
weighted avg       0.93      0.92      0.92      2420

[[593   0  12]
 [  3 794   0]
 [  4 185 829]]
accuracy is  0.915702479338843
```

```
forest = RandomForestClassifier(max_depth=2)
forest.fit(X1, y)
plot_decision_regions(X1, y, clf=forest, legend=2)
```



Soft Vector Machine : SVM

- SVM은 결정경계(최적의 라인, 초평면)를 만든다.
  - ✓ 초평면에 가장 가까이 있는 벡터를 지지벡터(Support Vector)라 한다.



그림출처) http://localhost:8888/notebooks/lab_vds/vds01_label_visualization.ipynb

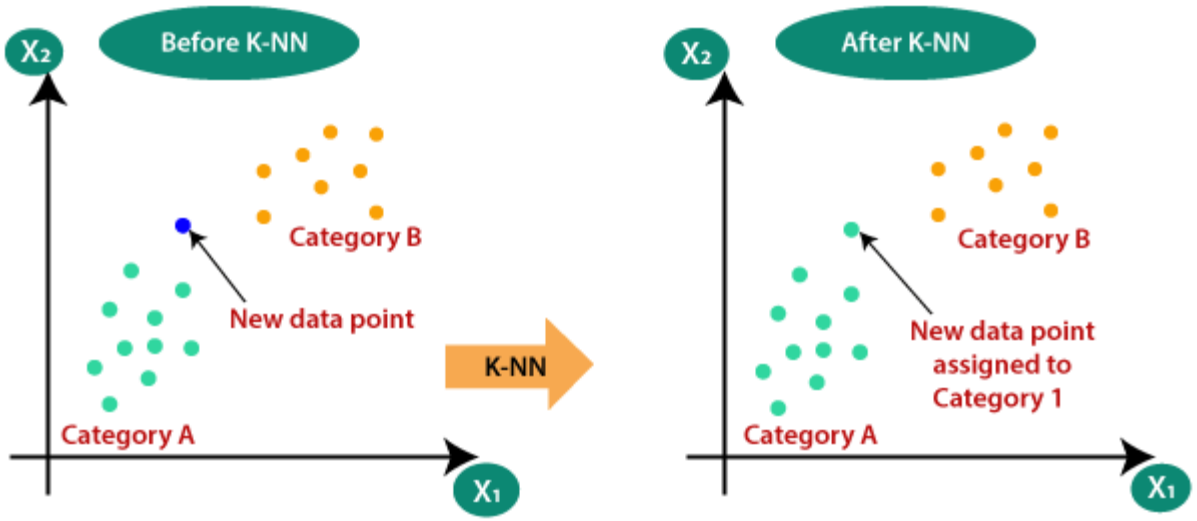
Soft Vector Machine

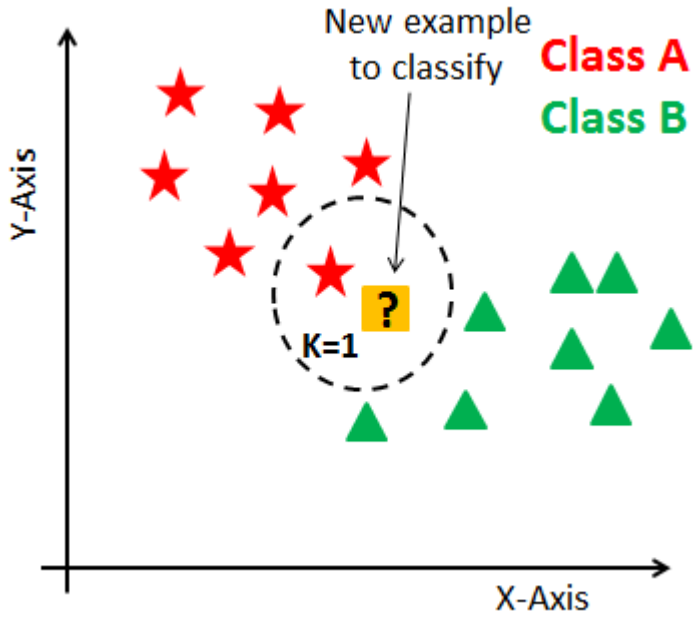
```
Model = SVC()
Model.fit(X_train, y_train)
y_pred = Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       600
           1       0.99      0.98      0.98       979
           2       0.95      0.99      0.97       841

    accuracy                           0.98      2420
   macro avg       0.98      0.98      0.98      2420
weighted avg       0.98      0.98      0.98      2420

[[580   0  20]
 [  0 957  22]
 [  3   8 830]]
accuracy is 0.978099173553719
```

K nearest neighbors : KNN

- KNN 알고리즘은 가까운 경계에 유사한 점이 있다고 가정한다. (유유상종 같은 개념)
  - ✓ KNN은 분류와 회귀에 둘 다 사용되지만, 분류에 더 많이 사용한다.





그림출처) http://localhost:8888/notebooks/lab_vds/vds01_label_visualization.ipynb

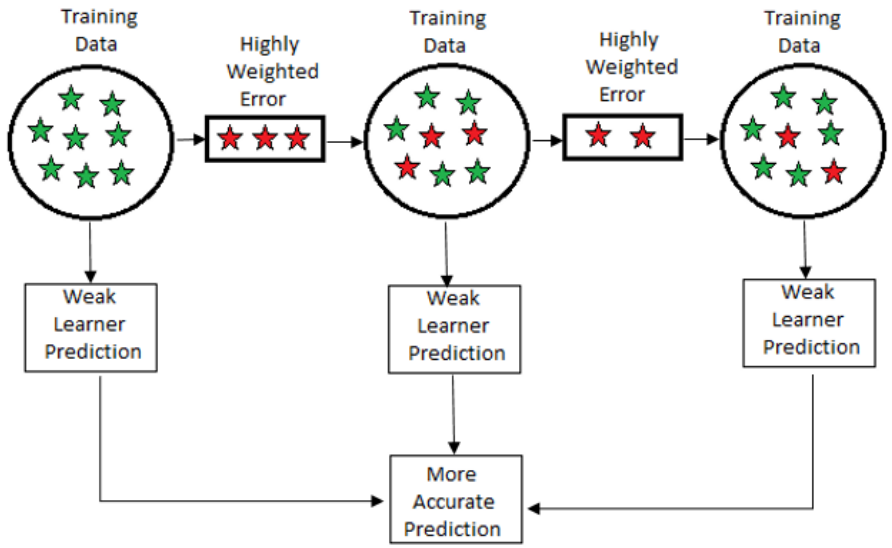
KNN

```python
# K-Nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier

Model = KNeighborsClassifier(n_neighbors=8)
Model.fit(X_train, y_train)
y_pred = Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       600
           1       0.98      0.95      0.96       979
           2       0.90      0.92      0.91       841

    accuracy                           0.94      2420
   macro avg       0.93      0.94      0.93      2420
weighted avg       0.94      0.94      0.94      2420

[[560   0  40]
 [  0 933  46]
 [ 44  23 774]]
accuracy is 0.9367768595041323
```

GradientBoosting

- Gradient Boosting은 약한 러너를 최적화 한다.
  - ✓ 따라서 Greedy 알고리즘으로 과적합 발생한다.




```python
from sklearn.ensemble import GradientBoostingClassifier
Model=GradientBoostingClassifier()
Model.fit(X_train,y_train)
y_pred=Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_pred,y_test))

#Accuracy Score
print('accuracy is ',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       600
           1       1.00      1.00      1.00       979
           2       1.00      1.00      1.00       841

    accuracy                           1.00      2420
   macro avg       1.00      1.00      1.00      2420
weighted avg       1.00      1.00      1.00      2420

[[600   0   0]
 [  0 979   0]
 [  0   0 841]]
accuracy is  1.0
```

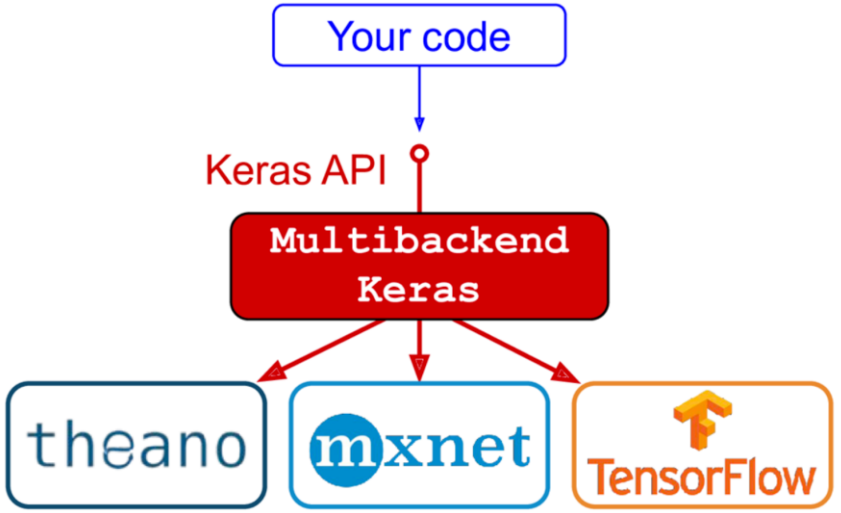
그림 출처) https://pub.towardsai.net/fully-explained-gradient-boosting-technique-in-supervised-learning-d3e293ca70e1

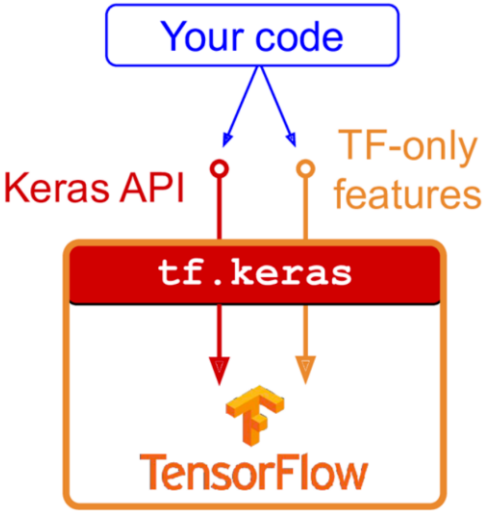
5) Tensorflow를 이용한 딥러닝 모델

Deep learning frameworks

- The most popular deep learning frameworks

tf.keras is extended Keras implementation based on TF





(source) https://developer.ibm.com/articles/compare-deep-learning-frameworks/

텐서플로우 2.0 시작하기: 초보자용 참고

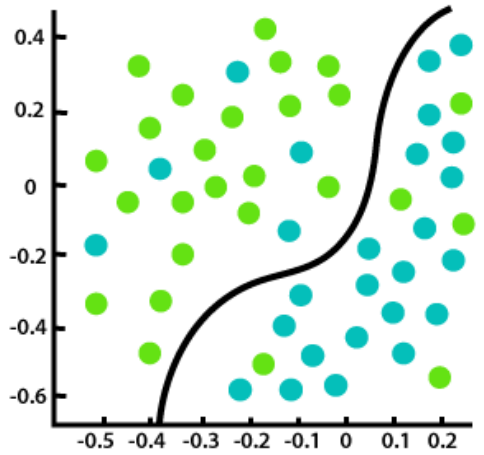
- 데이터셋 로드
- 모델 만들기
  - ✓ tf.keras.Sequential
- 모델은 훈련하기
  - ✓ model.compile
  - ✓ model.fit
- 모델 평가 하기
  - ✓ model.evaluate
- 모델 결과 가시화
  - ✓ plt.plot(history.history['loss'])

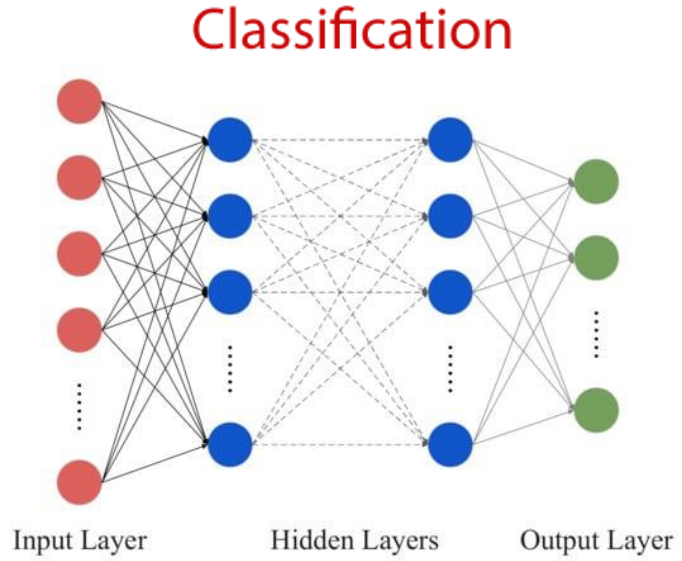
Classification MLPs

- *Typical classification MLP architecture*

Table 10-2. Typical classification MLP architecture

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross entropy	Cross entropy	Cross entropy

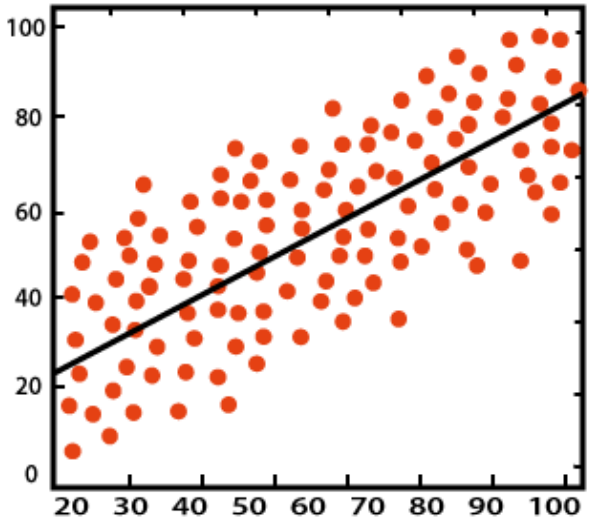




Regression MLPs

❖ Typical regression MLP architecture

Hyperparameter	Typical value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST)
# hidden layers	Depends on the problem, but typically 1 to 5
# neurons per hidden layer	Depends on the problem, but typically 10 to 100
# output neurons	1 per prediction dimension
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None, or ReLU/softplus (if positive outputs) or logistic/tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)



분류를 위한 데이터 라벨 전략 수립: 속도로 해보자!

```
num_classes = 3
class_labels= ['Jam', 'Slow', 'Normal']
```

```
def get_score(speed):
    if speed < 45:
        label = 'Jam'
    elif speed < 50:
        label = 'Slow'
    else :
        label = 'Normal'
    return label
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	OccRate	label
0	2017-04-02 0:00	43	34	9	0	50.3	1.90	Normal
1	2017-04-02 0:05	45	32	13	0	58.9	1.84	Normal

데이터를 특성과 라벨(숫자로 변환)

```
X = df[['ToVol','LaVol','MeVol','SmVol', 'Speed','OccRate']]
y = df['label']
```

```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

```
y
```

```
array([1, 1, 1, ..., 1, 1, 1])
```

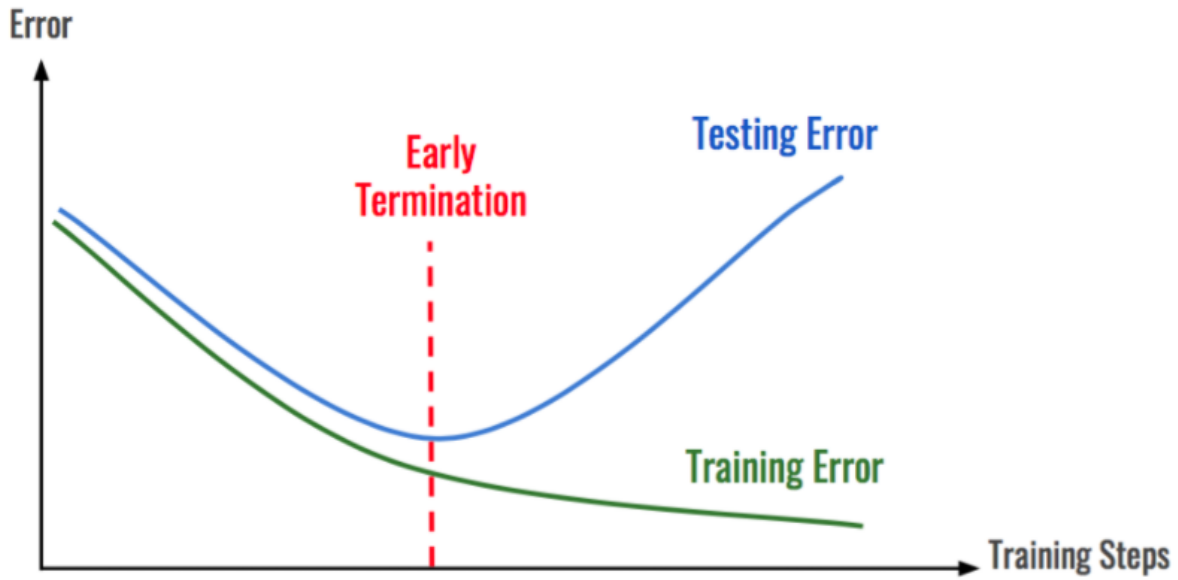
```
X[:3]
```

	ToVol	LaVol	MeVol	SmVol	Speed	OccRate
0	43	0	9	34	50.3	1.90
1	45	0	13	32	58.9	1.84
2	46	0	12	34	50.6	1.87

```
import numpy as np
from keras.utils import to_categorical
### Categorical data to be converted to numeric data
colors = [ "red", "green", "yellow", "red", "blue“ ]

### Universal list of colors
total_colors = [ "red", "green", "blue", "black", "yellow“ ]

### map each color to an integer
mapping = {}
for x in range(len(total_colors)):
  mapping[total_colors[x]] = x

# integer representation
for x in range(len(colors)):
  colors[x] = mapping[colors[x]]

one_hot_encode = to_categorical(colors)
print(one_hot_encode)
```

Sparse_Categoricalcrossentrpy를 사용함

Deep Learning 훈련과정 : 과적합 발생하면 조기종료한다.



Train/Test 데이터 나누기

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=1)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
(5644, 6) (5644,)
(2420, 6) (2420,)
```

```
X_train[:2]
```

```
array([[-0.67958742, -0.1033334 , -0.36417997, -0.7767931 ,  1.34731445,
        -0.52777298],
       [-1.55526491, -0.75688677, -1.46481163, -1.50330118,  0.66561375,
        -0.81414318]])
```

Deep Neural Network 모델 2개 만들어 보자

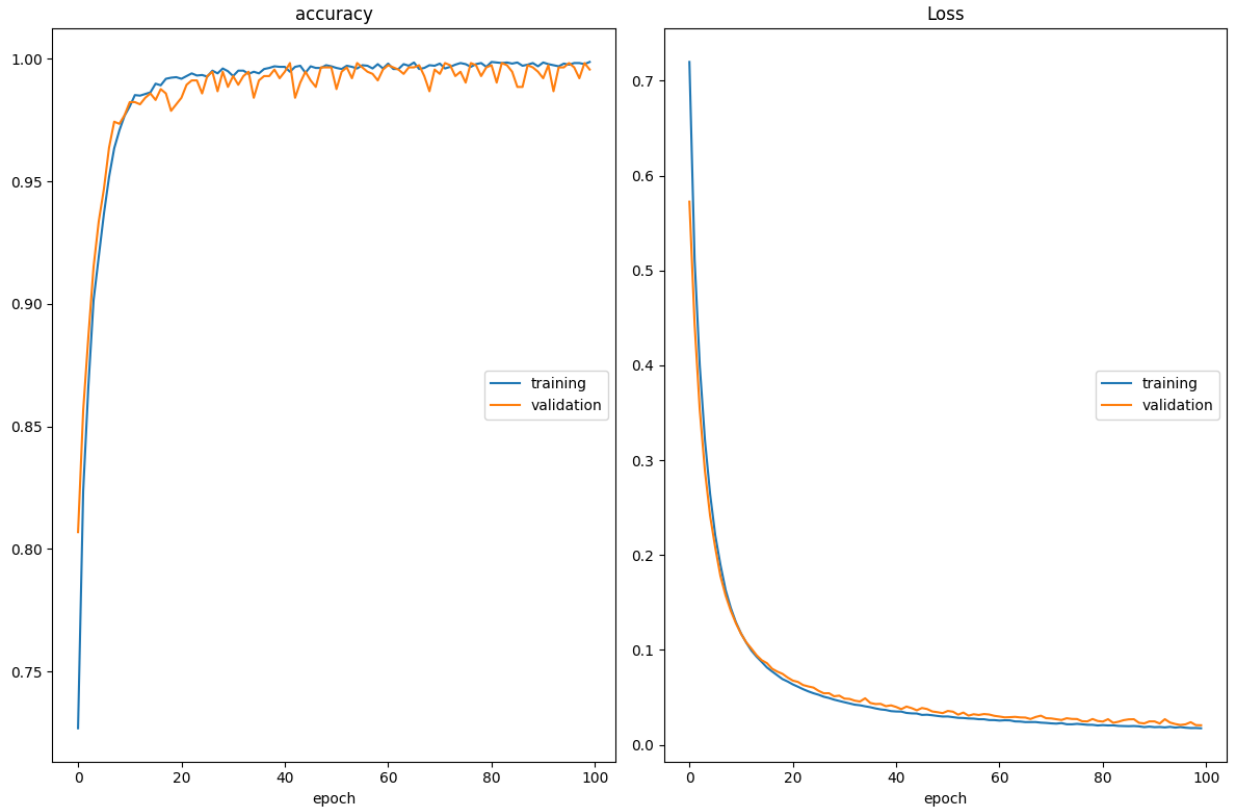
```
#import keras
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

from livelossplot import PlotLossesKeras
```

Shallow Deep learning

```
def model_shallow():
    model = Sequential([
        Dense(32, input_dim=X_train.shape[1], activation = 'relu'),
        Dense(num_classes, activation= 'softmax')
    ])
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

모델 훈련과정

```
model = model_shallow()
#history = model.fit(X_train,y_train, epochs=100, validation_split=0.2)
history = model.fit(X_train,y_train, epochs=100, validation_split=0.2,callbacks=[PlotLossesKeras()])
```

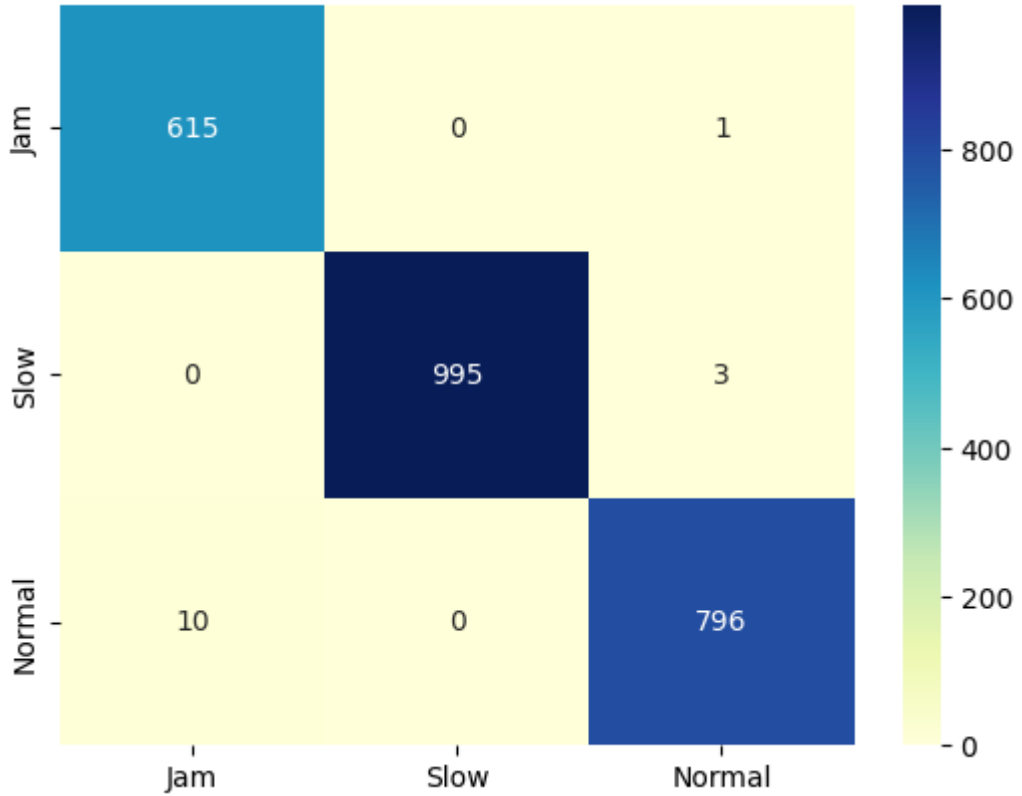



```
model.evaluate(X_test,y_test)
```

```
76/76 [==============================] - 0s 2ms/step - loss: 0.0187 - accuracy: 0.9942
[0.01872224360704422, 0.9942148923873901]
```

Confusion Matrix 분석

```
from sklearn.metrics import confusion_matrix
import numpy as np

y_pred_labels = [np.argmax(label) for label in y_pred]
cm = confusion_matrix(y_test,y_pred_labels)
sns.heatmap(cm , annot = True,  cmap='YlGnBu',
            fmt = 'd',xticklabels = class_labels,
            yticklabels = class_labels)
```

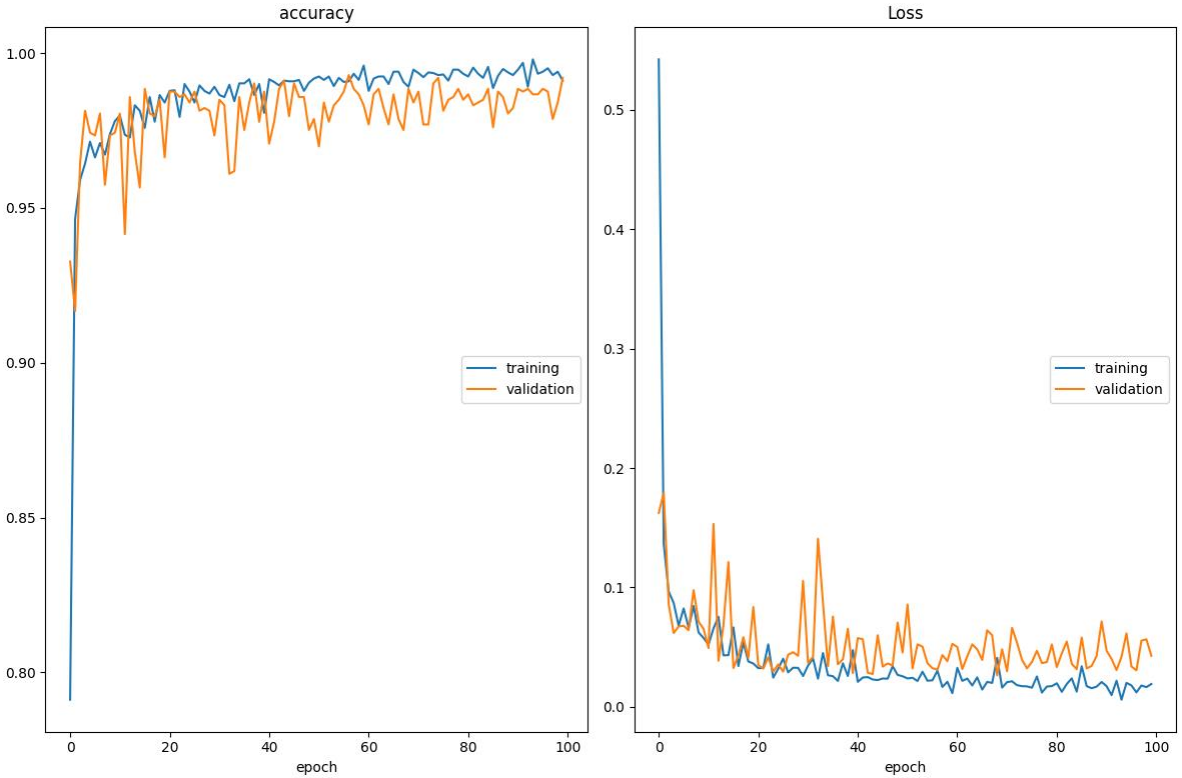


DNN : 깊은 은닉층

```
def model_deep():
    model = Sequential([
        Dense( 32, input_dim=X_train.shape[1], activation = 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( num_classes, activation= 'softmax')
    ])
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

```
model = model_deep()
history = model.fit(X_train,y_train, epochs=100, validation_split=0.2, callbacks=[PlotLossesKeras()])
```

DNN : 깊은 은닉층




```
model.evaluate(X_test, y_test)
```

```
76/76 [==============================] - 0s 2ms/step
```

```
[0.06313834339380264, 0.9851239919662476]
```

loss: 0.0631 - accuracy: 0.9851

Model_shallow2()

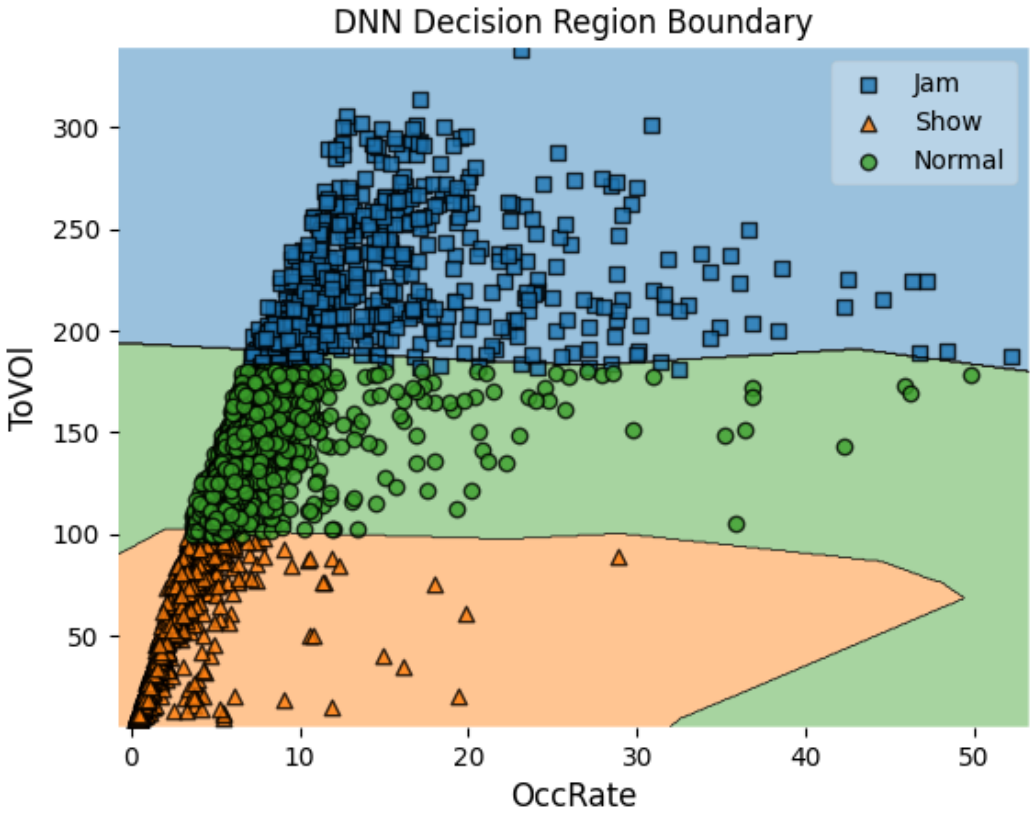
```
def model_shallow2():
    model = Sequential([
        Dense(16, input_dim=X_train.shape[1], activation = 'relu'),
        Dense(16, activation = 'relu'),
        Dense(num_classes, activation= 'softmax')
    ])
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

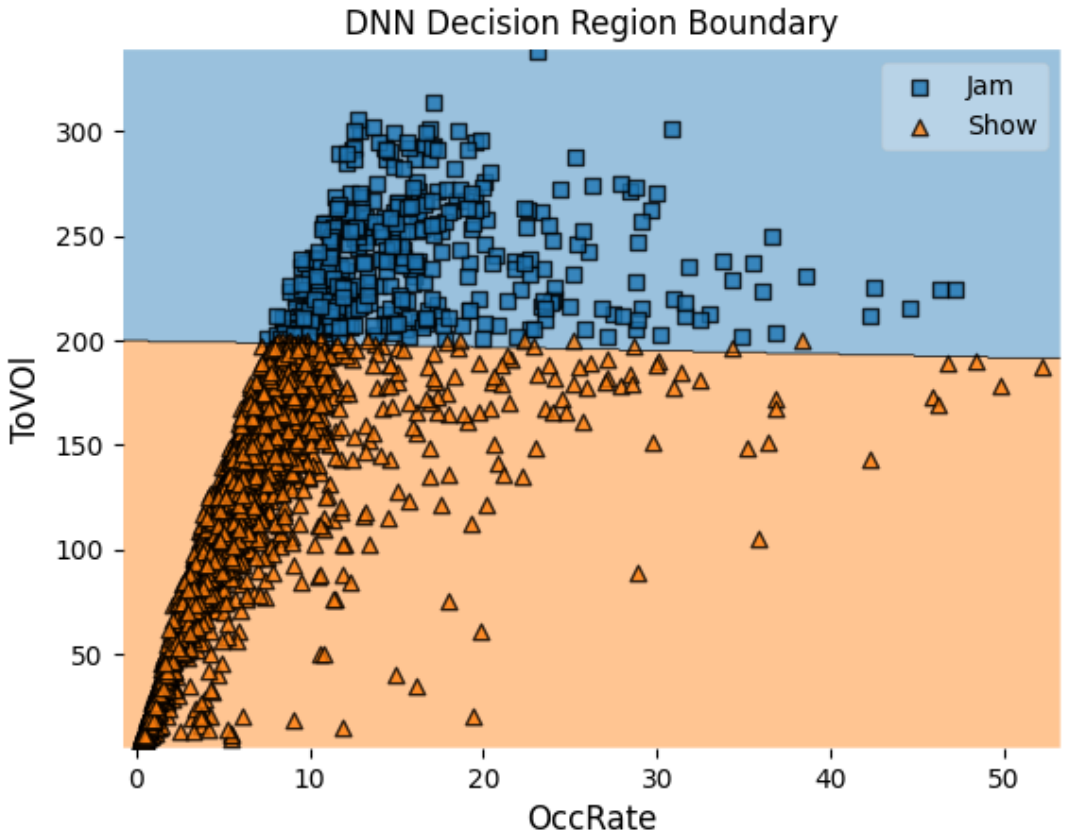
```
model = model_shallow2()
history = model.fit(X_train,y_train, epochs=100, validation_split=0.2, callbacks=[PlotLossesKeras()])
```

```
model.evaluate(X_test, y_test)
```

```
76/76 [==============================] - 0s 2ms/step - loss: 0.0152 - accuracy: 0.9942

[0.015153169631958008, 0.9942148923873901]
```

DNN 결정경계


DNN Decision Region Boundary
Jam
Show
Normal
ToVOI
OccRate
300
250
200
150
100
50
0
10
20
30
40
50


DNN Decision Region Boundary
Jam
Show
ToVOI
OccRate
300
250
200
150
100
50
0
10
20
30
40
50

6) 선형 회귀와 비선형 회귀 소개

회귀(Regression) 소개

❖ 회귀의 역사

✓ 영국의 통계학자 갈톤(Galton)의 유전적 특성중에 부모와 자식의 키 관계
- "사람의 키는 평균 키로 회귀(Regression)하려는 경향을 가진다는 자연의 법칙이 있다"

✓ 회귀 분석은 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법

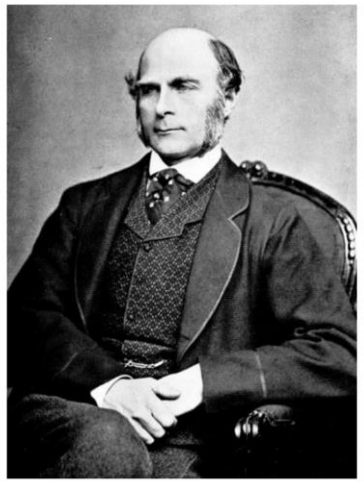
❖ 지도학습은 2가지 유형으로 나눔

✓ 회귀
- 연속적인 숫자 값

✓ 분류
- 예측값이 카테고리와 같은 이산형 클래스 값

❖ 선형회귀

✓ 실제 값과 예측 값의 차이를 최소화하는 직선형 회귀
- 주택 가격이 주택의 크기로만 결정

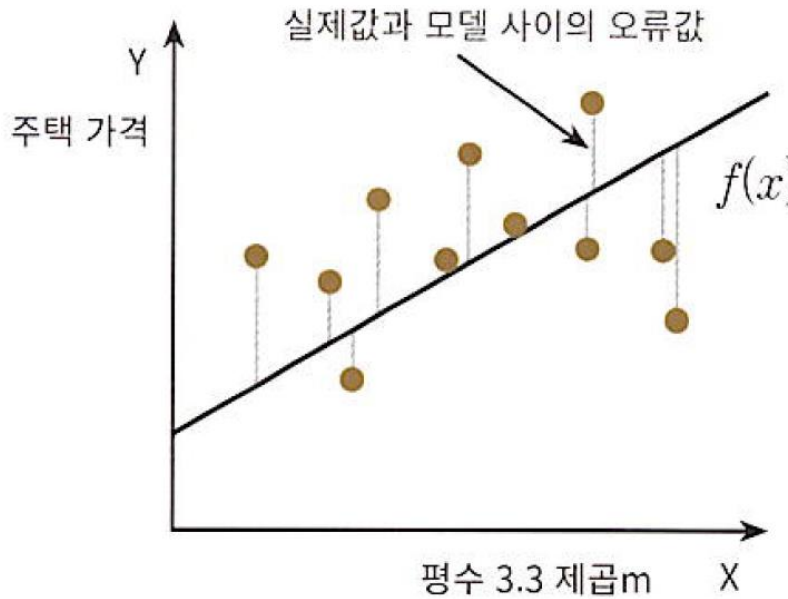


Sir Francis Galton (1822 ~ 1911)

회귀(Regression) 소개

❖ 단순 선형회귀

✓ 1개의 독립변수, 1개의 종속변수



$$f(x) = w_0 + w_1 * x$$

RSS = (#1 주택 가격 – (w_0+w_1 #1 주택크기)^2

+ (#2 주택 가격 – (w_0+w_1 #2 주택크기)^2

+ (#3 주택 가격 – (w_0+w_1 #3 주택크기)^2

+ … (모든 학습 데이터에 대해 RSS 수행)

$$\mathrm{RSS}(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_o + w_1 \times x_i))^2$$

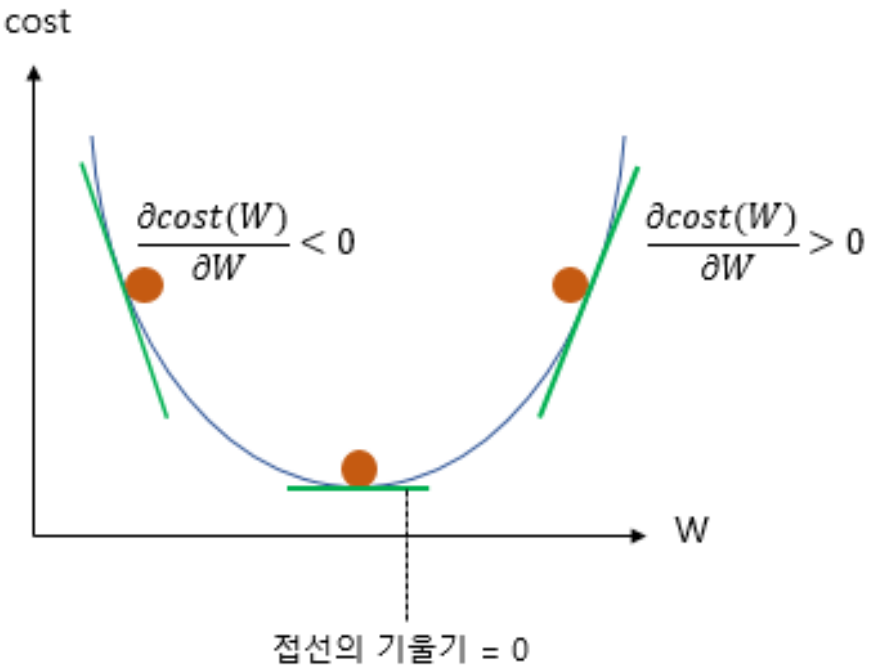
옵티마이저 : 경사하강법(Gradient Descent)

❖ **옵티마이저(Optimizer) 또는 최적화 알고리즘**

✓ 머신 러닝 학습은 결국 비용 함수를 최소화하는 가중치, 편향을 찾기 위한 작업을 수행

$$w_{\text{updated}} \approx w - \alpha \cdot \frac{\partial \text{cost}}{\partial w}$$

$$W, b \rightarrow minimize\ cost(W, b)$$

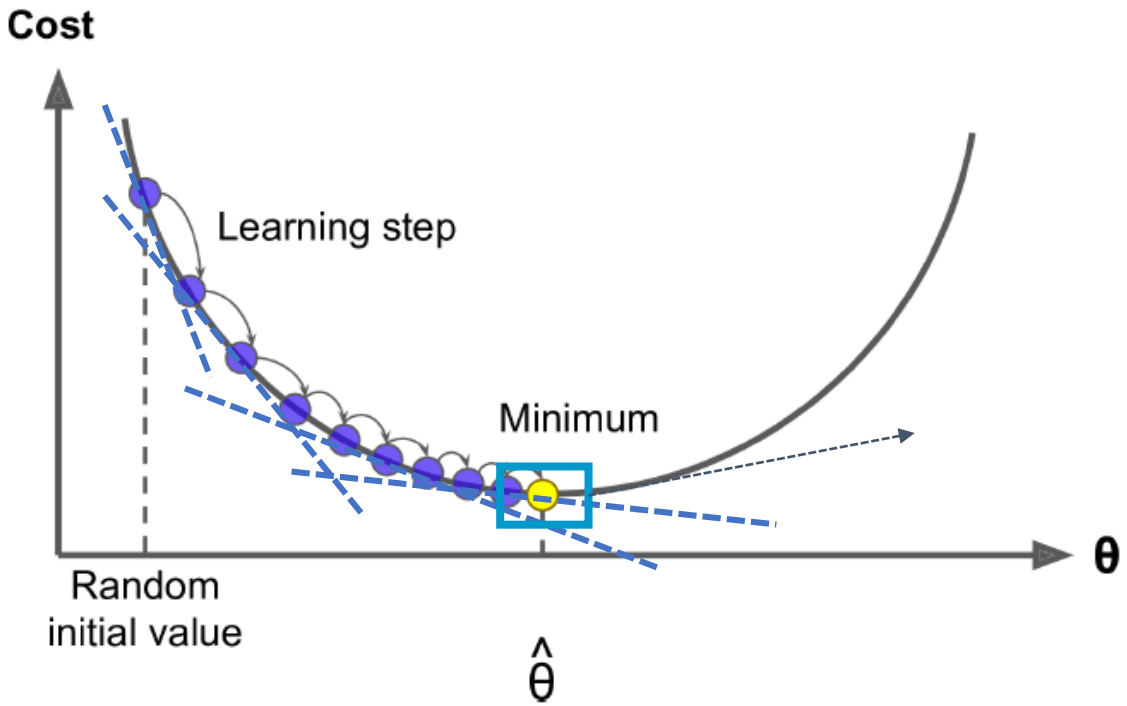


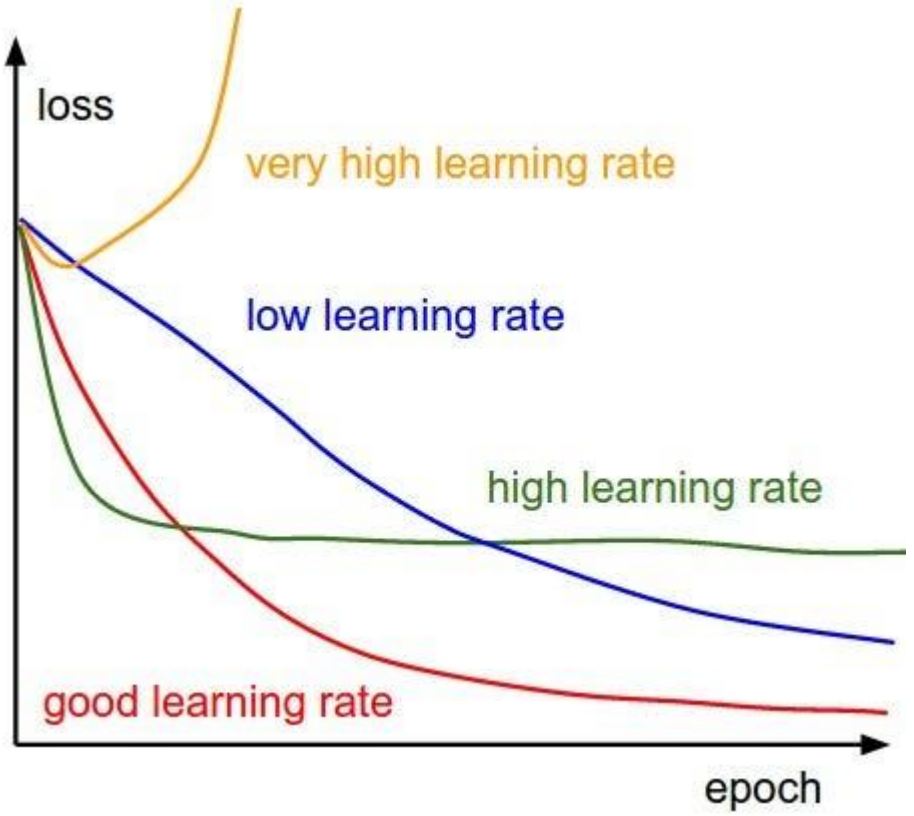
경사 하강법 및 학습률 이해

경사 하강법 (Gradient Descent)

$\eta \sim \text{learning rate}$

$$w = w - \eta \frac{2}{N} \sum_{i=1}^{N} (\text{real}_i - \text{pred}_i)$$





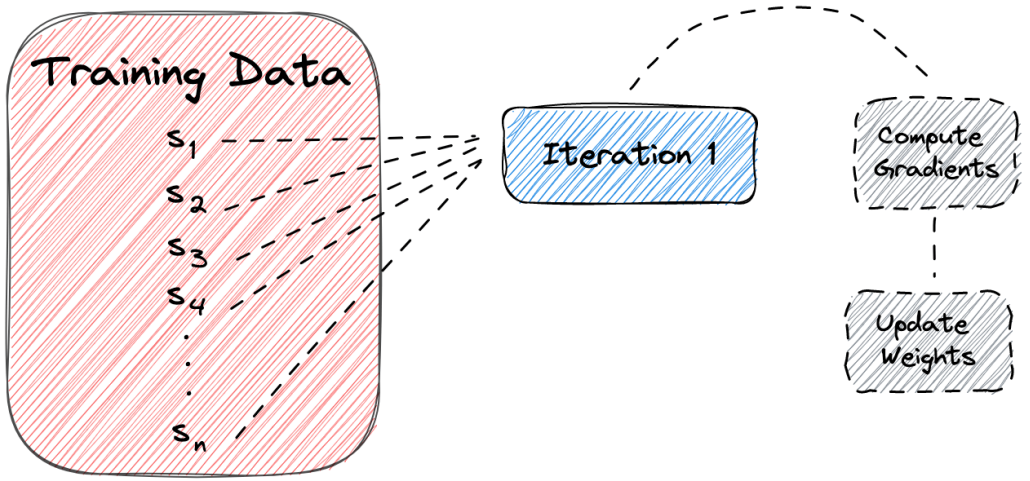
https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de

배치와 이포크(Epoch) 개념

- 전체 학습 데이터를 한 번씩 사용 한 것을 '에폭'이라 함
- 훈련 데이터 묶음을 배치라고 한다.

https://otugi.tistory.com/m/350

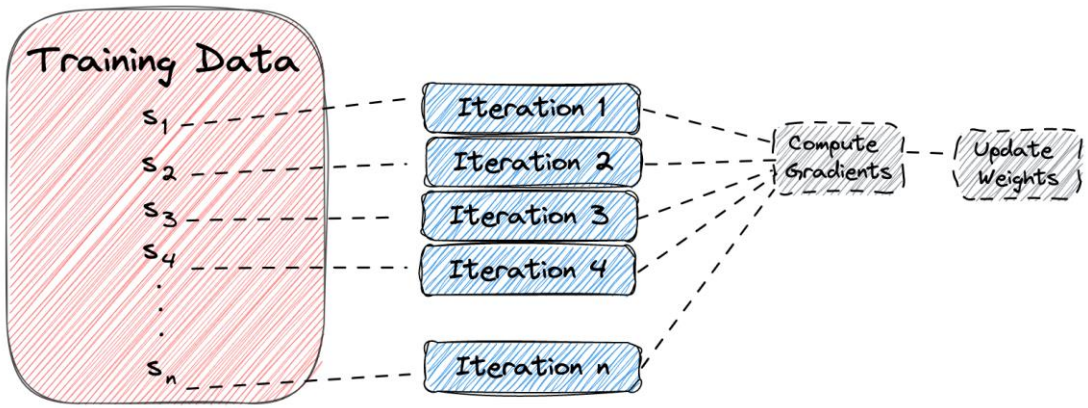
Batch Gradient



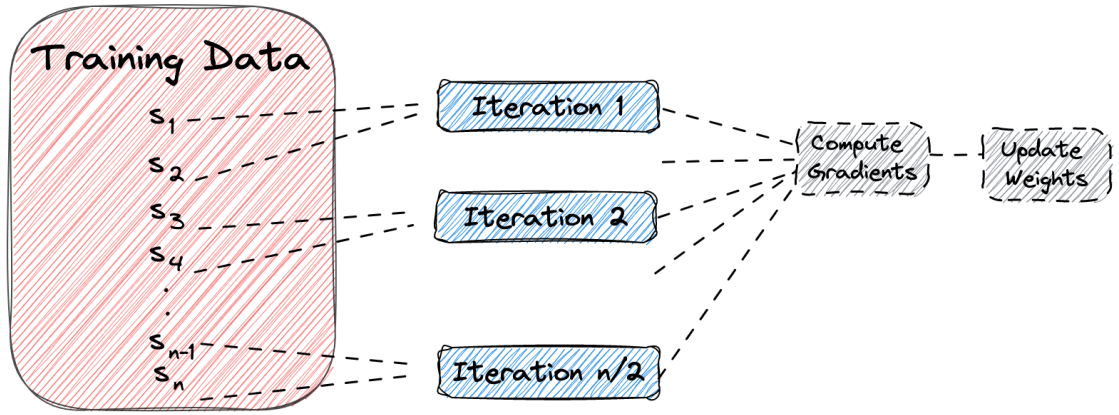
(source) https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch#1-batch-gradient-descent

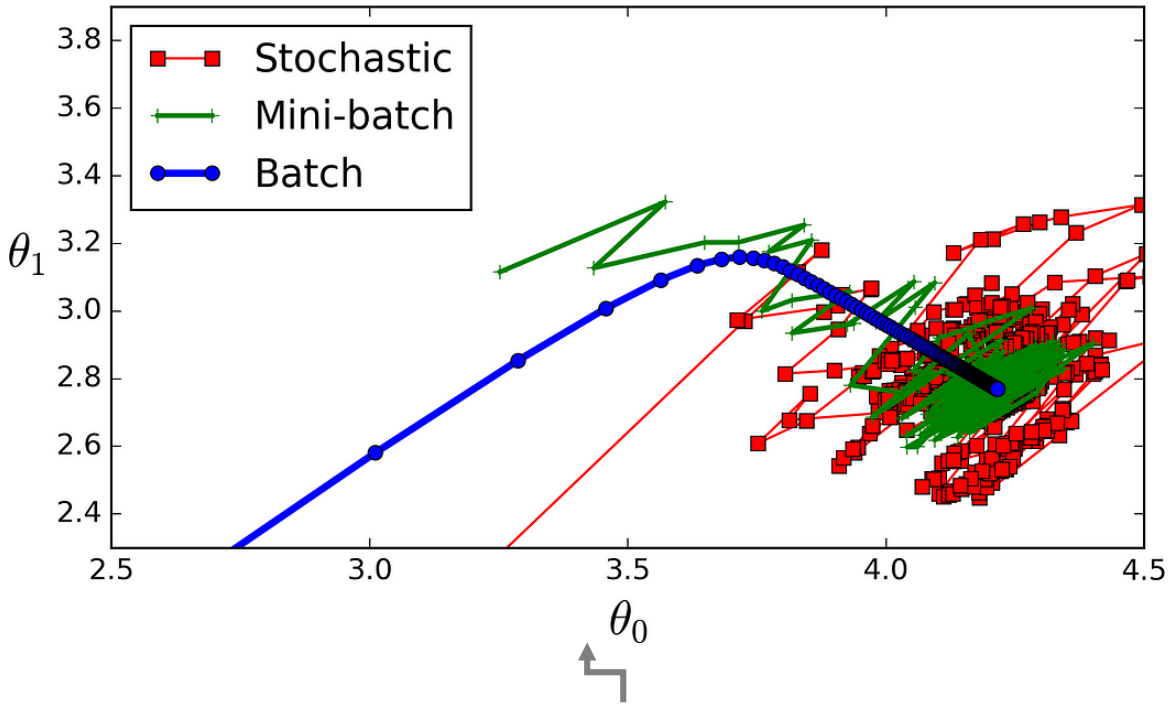
Types of Gradient Descent

Stochastic Gradient



Mini-Batch Gradient





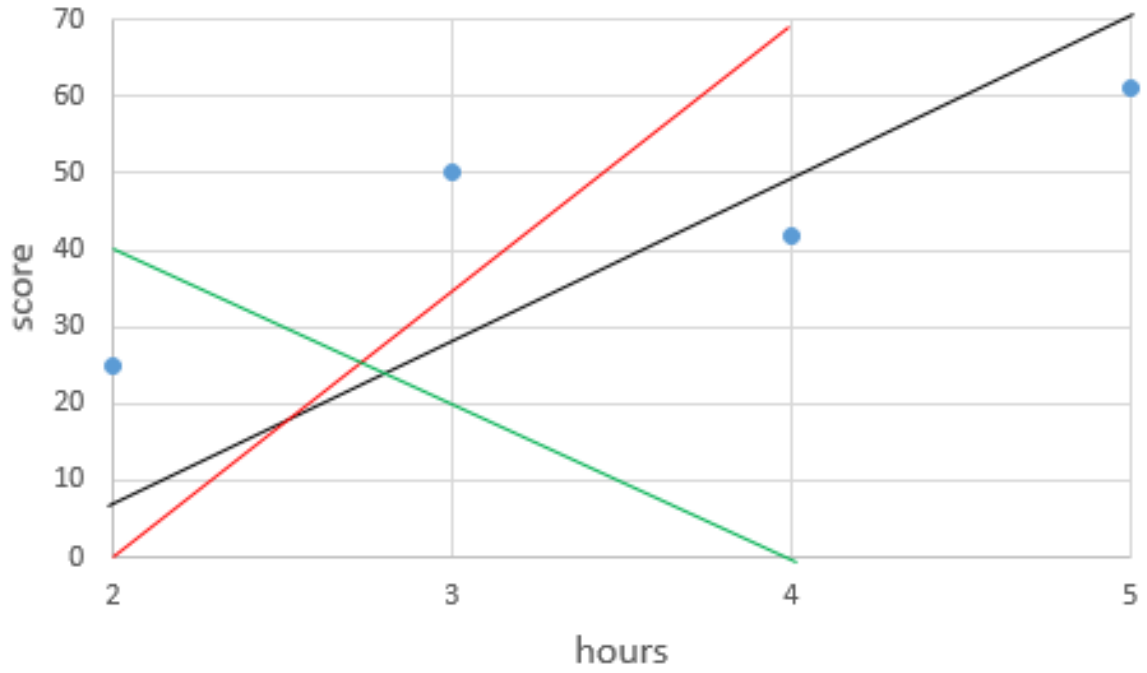
(source) https://sweta-nit.medium.com/batch-mini-batch-and-stochastic-gradient-descent-e9bc4cacd461

(source) https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch#1-batch-gradient-descent

선형 회귀 : 가설(Hypothesis)

❖ 어떤 학생의 공부 시간에 따라서 얻은 점수 데이터

hours(x)	score(y)
2	25
3	50
4	42
5	61



$$H(x) = Wx + b$$

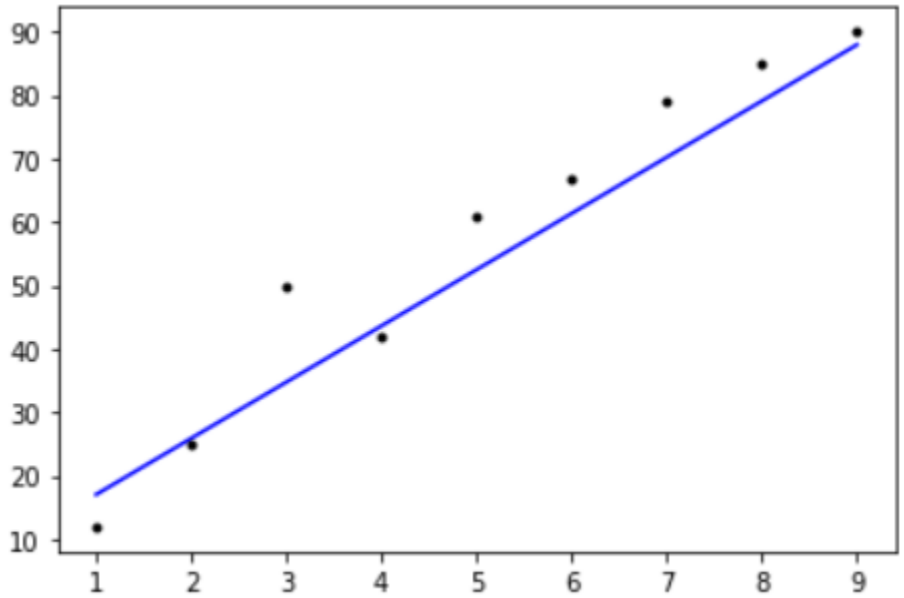
케라스로 구현하는 선형 회귀

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers
```

```
X=np.array([1,2,3,4,5,6,7,8,9])
# 공부하는 시간
y=np.array([12,25,50,42,61, 67, 79, 85, 90])
```

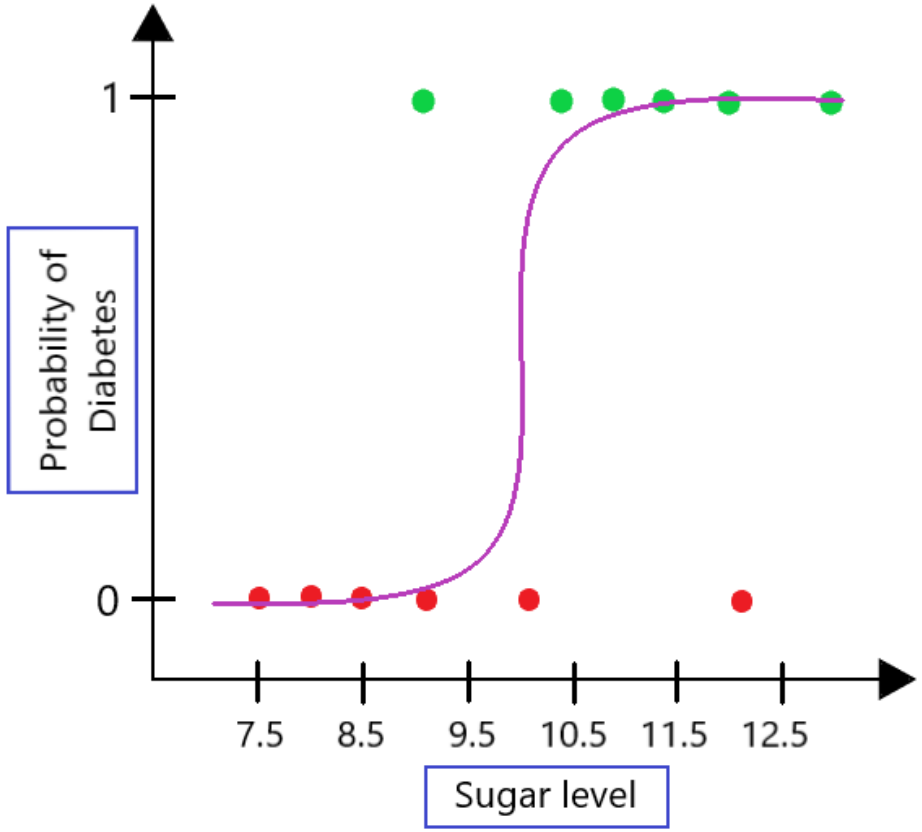
```
model = Sequential()

model.add(Dense(1, input_dim=1, activation='linear'))
sgd = optimizers.SGD(lr=0.01)
model.compile(optimizer=sgd ,loss='mse',metrics=['mse'])
history=model.fit(X,y, batch_size=1, epochs=30, shuffle=False)
```

```
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```

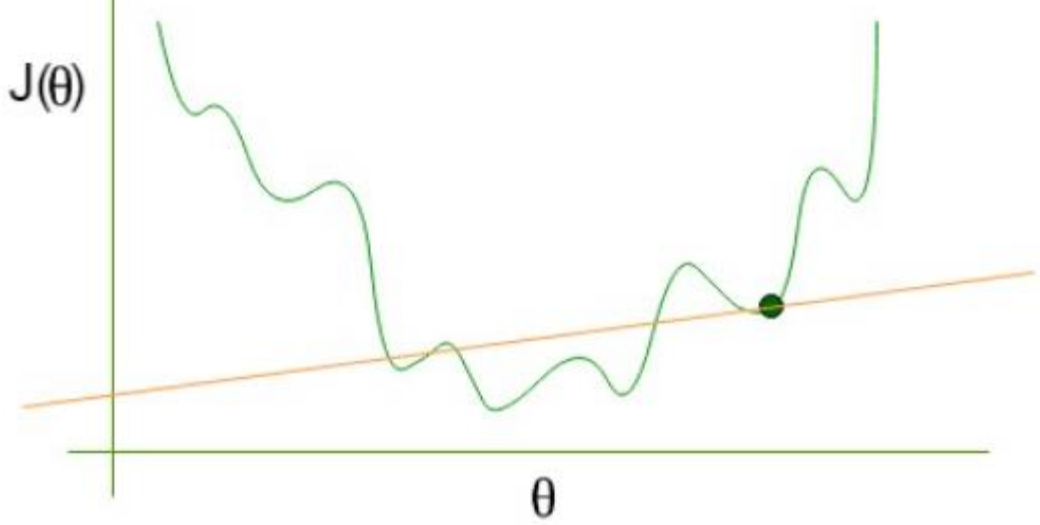


비선형 회귀 문제

이진 분류(Binary Classification): 둘 중 하나를 결정하는 문제
이런 문제를 풀기 위한 알고리즘으로 로지스틱 회귀(Logistic Regression)



비용함수는 볼록한 모양이 아니고,
여러 개의 로컬미니엄이 있는 모양이다.



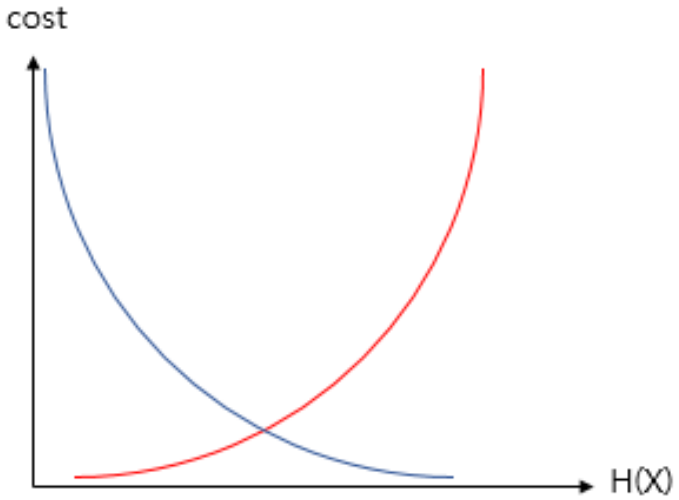
https://www.geeksforgeeks.org/ml-cost-function-in-logistic-regression/

Cross Entropy

분류 평가 지표로 Log Loss (Cross Entropy)를 고려하자.

$$J(W) = -\frac{1}{n}\sum_{i=1}^{n}[y^{(i)}logH(x^{(i)}) + (1-y^{(i)})log(1-H(x^{(i)}))]$$

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} cost\left(H(x^{(i)}), y^{(i)}\right)$$

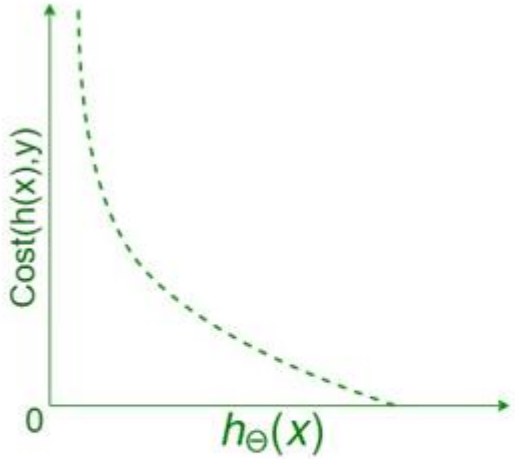


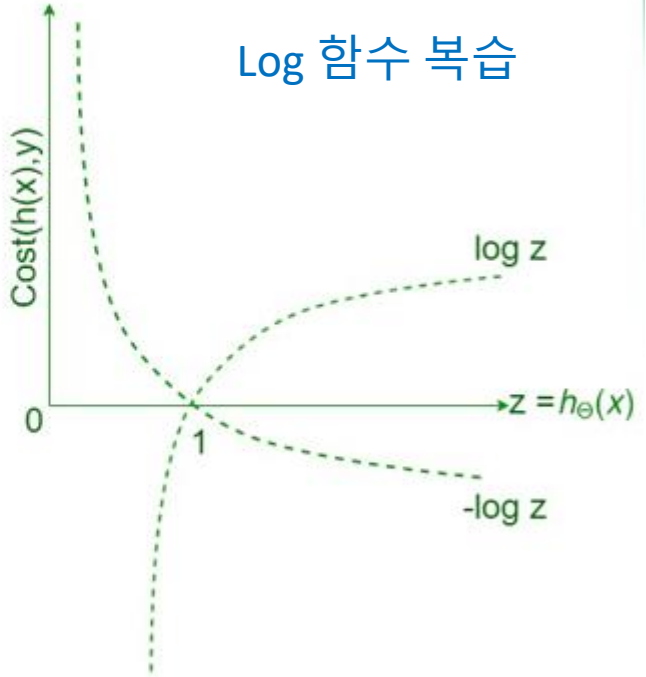
$$\text{cost}\,(H(x), y) = -[ylogH(x) + (1-y)log(1-H(x))]$$

$$H(X) = \frac{1}{1+e^{-(Wx+b)}} = \sigma(Wx+b)$$

로지스틱 회귀의 비용함수

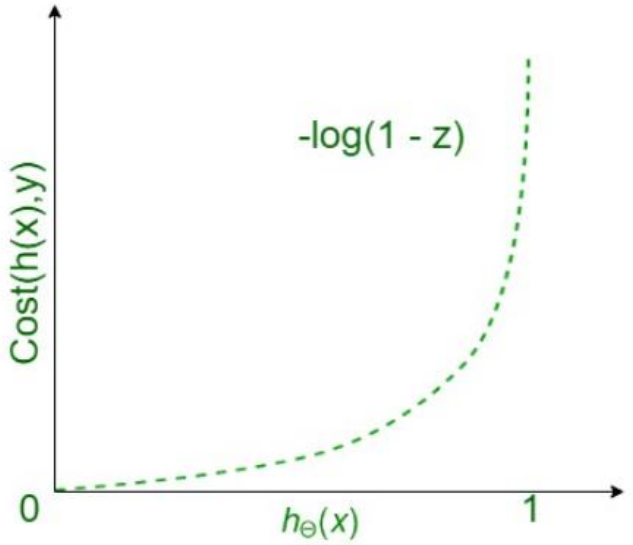
$$\text{if } y = 0 \rightarrow \text{cost}\,(H(x), y) = -\log(1 - H(x))$$





Log 함수 복습

$$\text{if } y = 1 \rightarrow \text{cost}\,(H(x), y) = -\log(H(x))$$



https://www.geeksforgeeks.org/ml-cost-function-in-logistic-regression/

회귀를 위한 기울기 경사 문제

선형회귀 최적화 과정

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \qquad h_\theta(x) = \theta^\top x$$

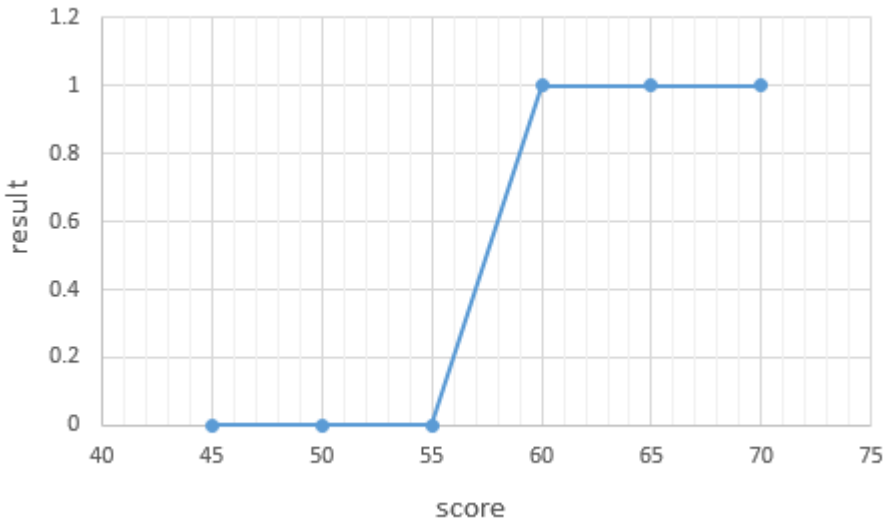
비선형회귀 최적화 과정

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \qquad h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

비선형회귀 예제: 이진 분류(Binary Classification)

❖ 학생들이 시험 성적에 따라서 합격, 불합격 데이터

score(x)	result(y)
45	불합격
50	불합격
55	불합격
60	합격
65	합격
70	합격




```
X=np.array([-50, -40, -30, -20, -10, -5, 0, 5, 10, 20, 30, 40, 50])
y=np.array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
```

Tensorflow 2.x로 시작하는 로지스틱 회귀

```
import numpy as np
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers
```

```
model=Sequential()
```

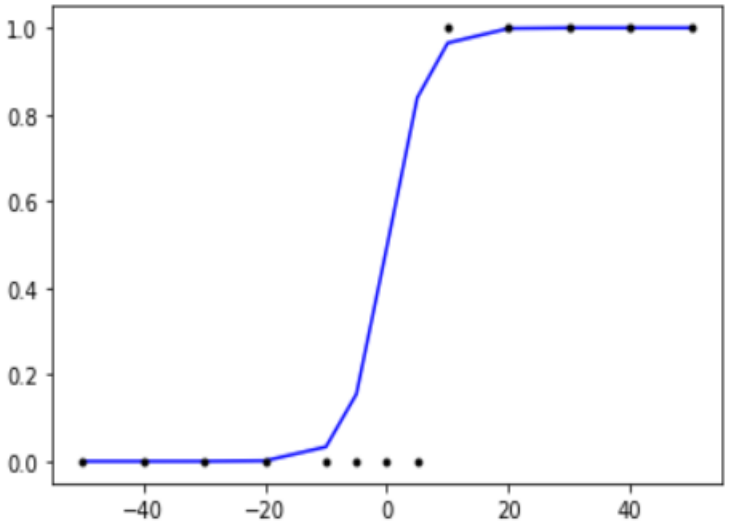
```
model.add(Dense(1, input_dim=1, activation='sigmoid'))
```

```
sgd=optimizers.SGD(lr=0.01)
```

```
model.compile(optimizer=sgd ,
              loss='binary_crossentropy',metrics=['binary_accuracy'])
```

```
history = model.fit(X,y, epochs=20, shuffle=False)
```

```
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```



7) 딥러닝 교통 회귀 문제

코랩에서 github 데이터 읽어오기

```
import pandas as pd
```

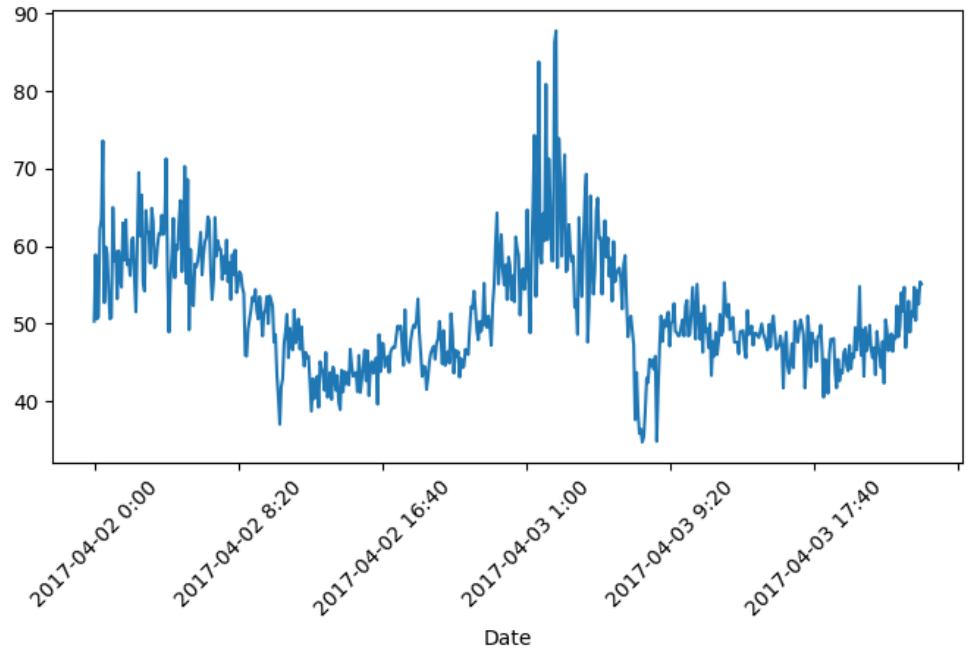
```
df = pd.read_csv('https://raw.githubusercontent.com/hongsukyi/Lectures/main/vds/vds.csv')
```

```
df.set_index('Date', inplace=True)
```

```
df.head(2)
```

Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
2017-04-02 0:00	43	34	9	0	50.3	1.90
2017-04-02 0:05	45	32	13	0	58.9	1.84

```
df['Speed'][:576].plot(rot=45,figsize=(8,4))
```



Multivariate Regression

```
features = ['ToVol','LaVol','MeVol','SmVol','Occ.Rate']
X = df[features]
y = df.iloc[:,4:5].values
```

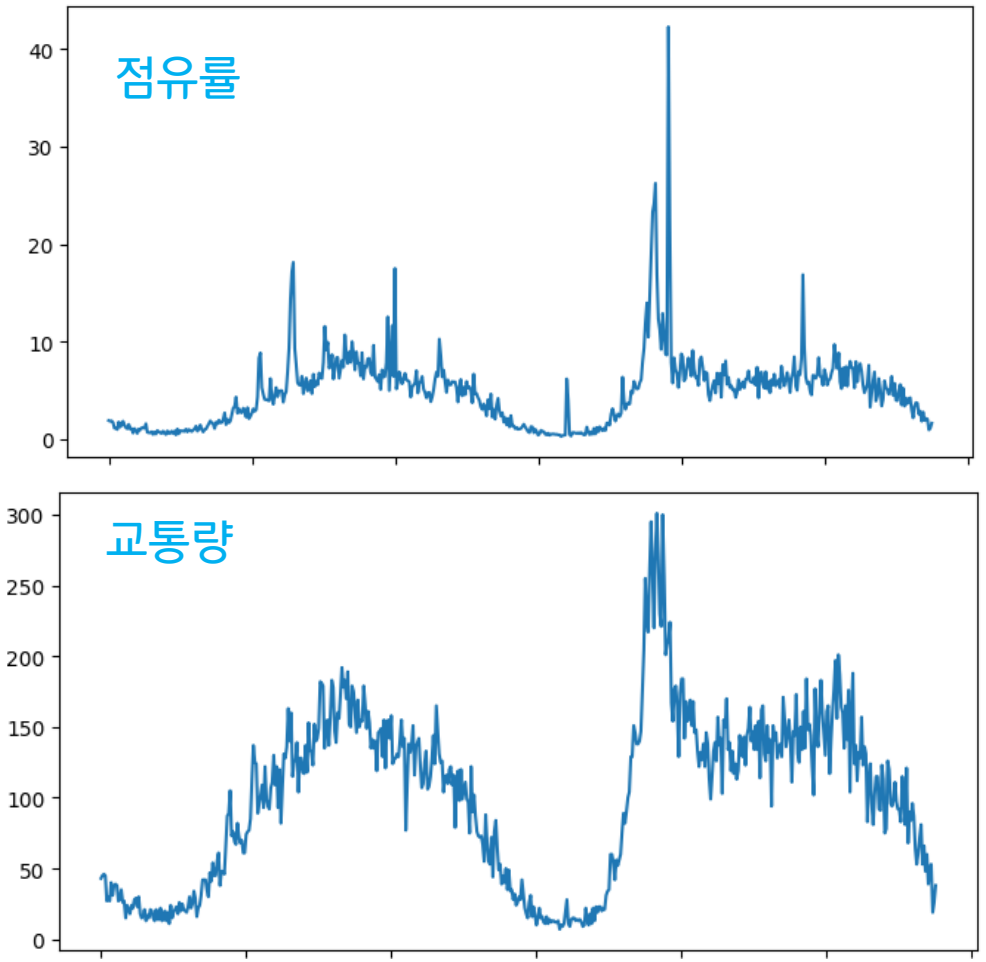
```
print(y)
```

```
[[50.3]
 [58.9]
 [50.6]
 ...
 [50.6]
 [59.3]
 [52.5]]
```

```
max_speed = y.max(); max_speed
```

```
87.8
```

```
df['ToVol'][:576].plot(rot=45,figsize=(8,4))
```



정규화

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
#scaler = StandardScaler()
scaler = MinMaxScaler()
scaled_X = scaler.fit_transform(X)
scaled_y = scaler.fit_transform(y)
```

```
print(scaled_X)
print(scaled_y)
```

```
[[0.11144578 0.         0.06206897 0.12903226 0.02039819]
 [0.11746988 0.         0.08965517 0.12096774 0.01966532]
 [0.12048193 0.         0.08275862 0.12903226 0.02003176]
 ...
 [0.07831325 0.         0.02758621 0.10483871 0.01380237]
 [0.0753012  0.         0.06896552 0.0766129  0.01429095]
 [0.09939759 0.         0.04137931 0.125      0.01844387]]
[[0.52350699]
 [0.63278272]
 [0.52731893]
```

Train과 test를 8:2로 분할

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(scaled_X, scaled_y,
                                                    test_size=0.2, shuffle=False)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
(6451, 5) (6451, 1)
(1613, 5) (1613, 1)
```

```
num_features = len(X_train[1])
print('number of features :', num_features)
```

→ number of features : 5

MLP 회귀모델

```
def dnn_reg1():
    model = Sequential([
        Dense(n1, activation='relu', input_shape=[num_features]),
        Dense(1)
    ])
    model.compile(optimizer='adam',loss= 'mse', metrics=['mae'])
    return model
```

```
n1 = 32
n2 = 32
n3 = 32
n4 = 32
```

```
def dnn_reg2():
    model = Sequential([
        Dense(n1, activation='relu',
              input_shape=[num_features]),
        Dense(n2, activation='relu'),
        Dense(1)
    ])
```

```
def dnn_reg4():
    model = Sequential([
        Dense(n1, activation='relu',
              input_shape=[num_features]),
        Dense(n2, activation='relu'),
        Dense(n3, activation='relu'),
        Dense(n4, activation='relu'),
        Dense(1)
    ])
```

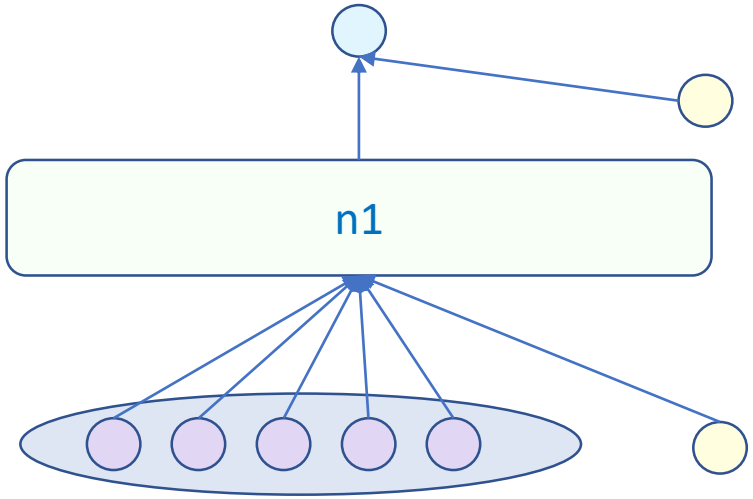
회귀 모델 아키텍처 및 파라미터 구하기

```
def dnn_reg1():
    model = Sequential([
        Dense(n1, activation='relu', input_shape=[num_features]),
        Dense(1)
    ])
```

```
model = dnn_reg1()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                192

 dense_1 (Dense)             (None, 1)                 33

=================================================================
```

파라미터 개수 계산하기
Dense_1 (32+1)*1=33
Dense: (5+1)*32=192



X=5개

bias=1개

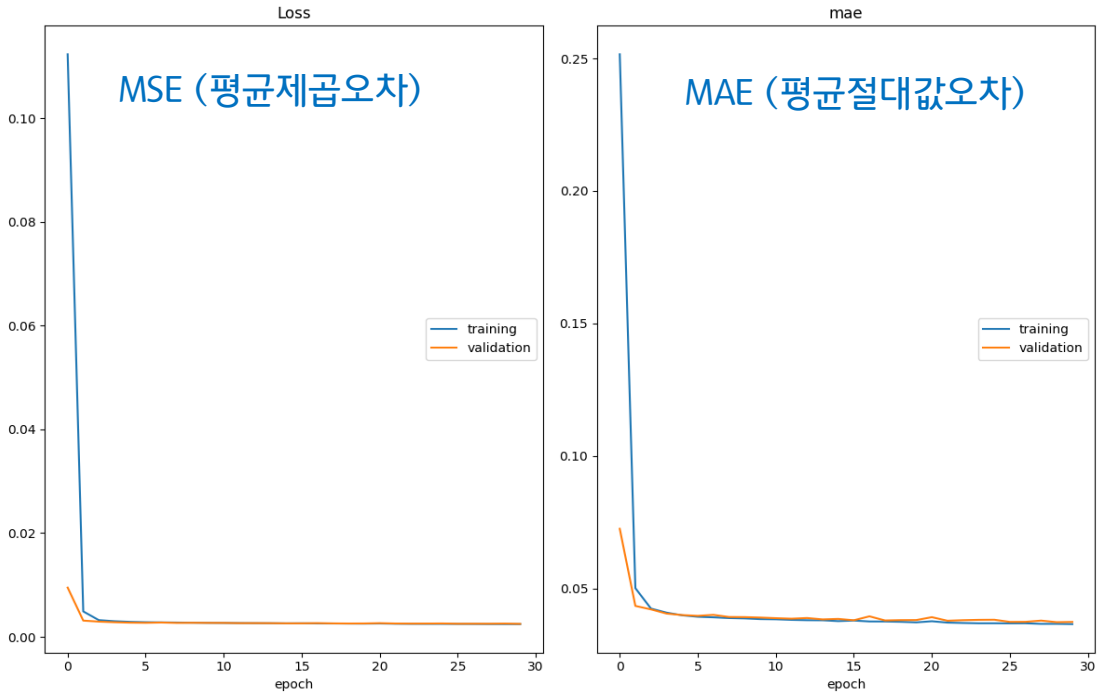
모델 훈련 및 실시간 가시화

Colab에서 꼭 미리 실행이 필요합니다.

!pip install livelossplot

```
from livelossplot import PlotLossesKeras
```

```
history = model.fit(X_train,y_train, epochs=30, validation_split=0.2, batch_size=32, callbacks=[PlotLossesKeras()])
```




```
# 테스트 데이터 세트로 모델 평가: Loss 값 확인
mse, mae = model.evaluate(X_test, y_test, verbose=0)
```

```
Neuron1= 32,  MSE=  0.0025, MAE=  0.0362
```

훈련과정 Loss와 MAE: history

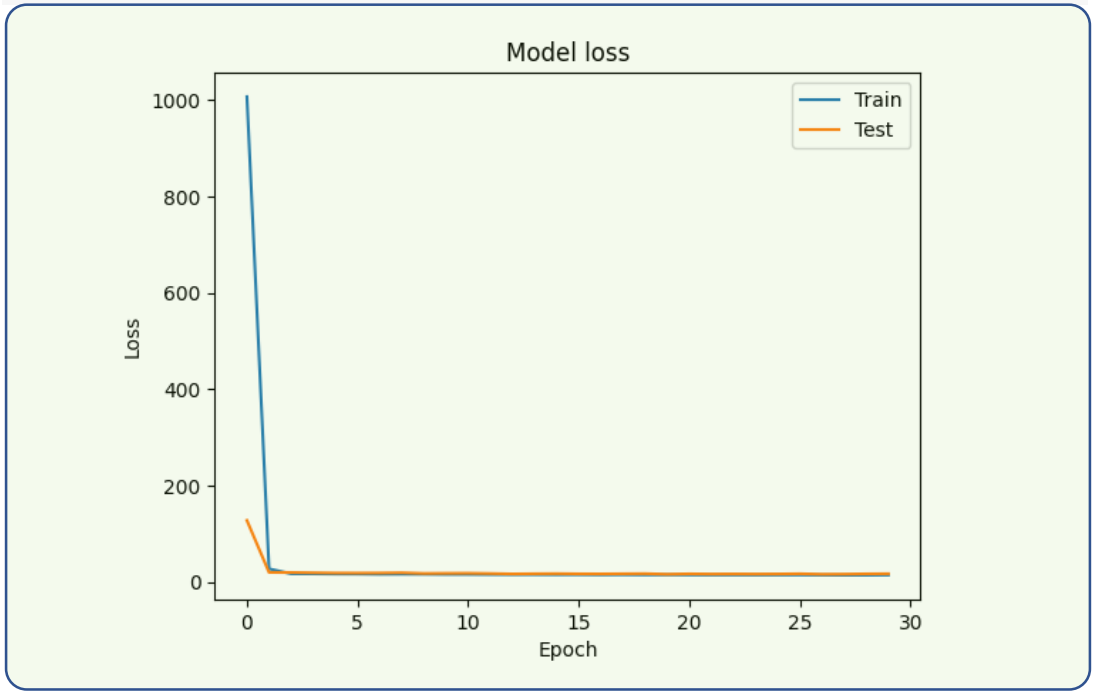
```
history.history.keys()
```

```
dict_keys(['loss', 'mae', 'val_loss', 'val_mae'])
```

```
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail()
```

	loss	mae	val_loss	val_mae	epoch
25	14.801393	2.847822	17.192307	2.992428	25
26	14.521421	2.806103	16.137362	2.960289	26
27	14.246248	2.754538	16.351469	2.987771	27
28	14.358523	2.774629	16.986986	2.998828	28
29	14.627845	2.818771	17.263212	3.201360	29

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='upper right')
```



이제 DNN 모델로 예측: 속도

표준화(정규화)로 스케일 상태 → 원래 속도로 환원해주자

```
y_pred = model.predict(X_test, verbose=0)
print(y_pred.shape)
```

```
(1613, 1)
```

```
print('(scaled) predictions : \n', y_pred[:2])
print('\n (scaled) actural values :\n',y_test[:2])
```

```
(scaled) predictions :
 [[0.4665929]
 [0.4981718]]

 (scaled) actural values :
 [[0.50190597]
 [0.54002541]]
```

```
y_inv_pred = scaler.inverse_transform(y_pred)
y_inv_test = scaler.inverse_transform(y_test)
```

```
print('predictions : \n', y_inv_pred[:2])
print('\nactural values :\n',y_inv_test[:2])
```

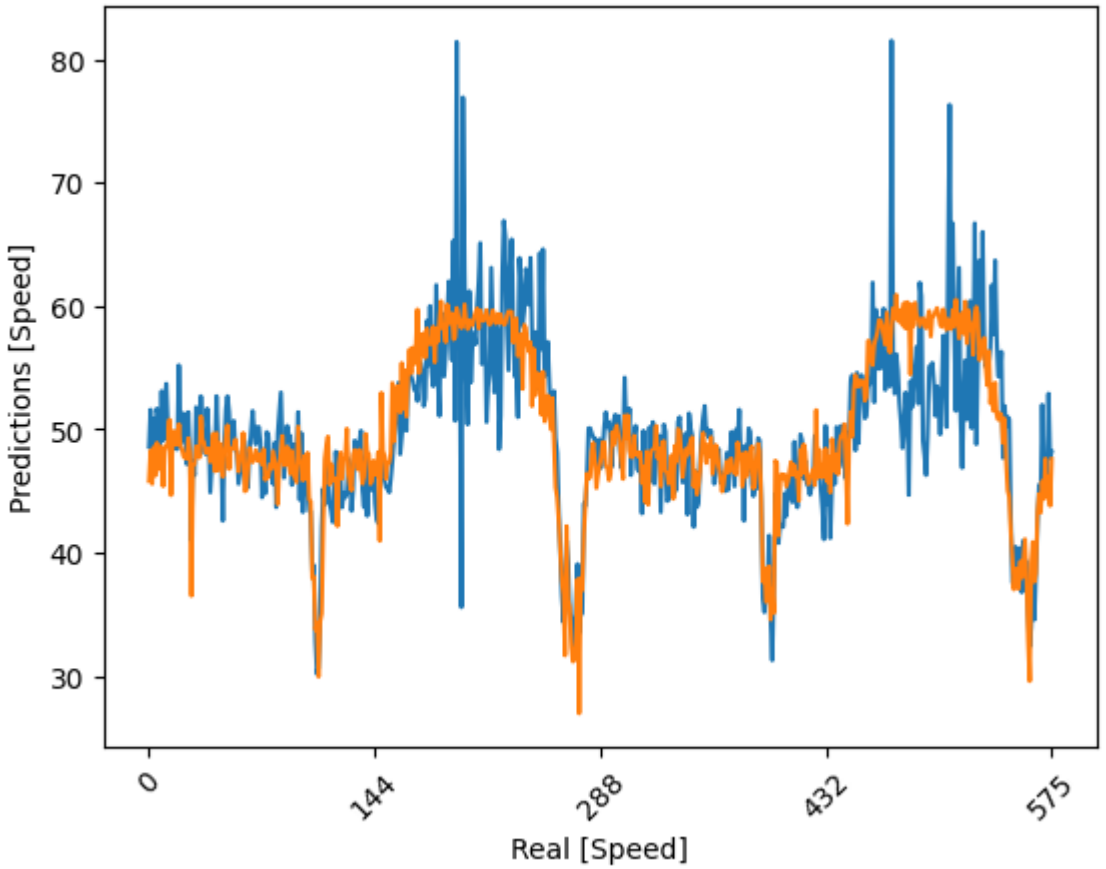
```
predictions :
 [[45.82086]
 [48.30612]]

actural values :
 [[48.6]
 [51.6]]
```

DNN으로 2일간 속도 예측

```
# 가시화를 위해서 2차원을 1차원으로
test_prediction = y_inv_pred.flatten()
test_real = y_inv_test.flatten()
```

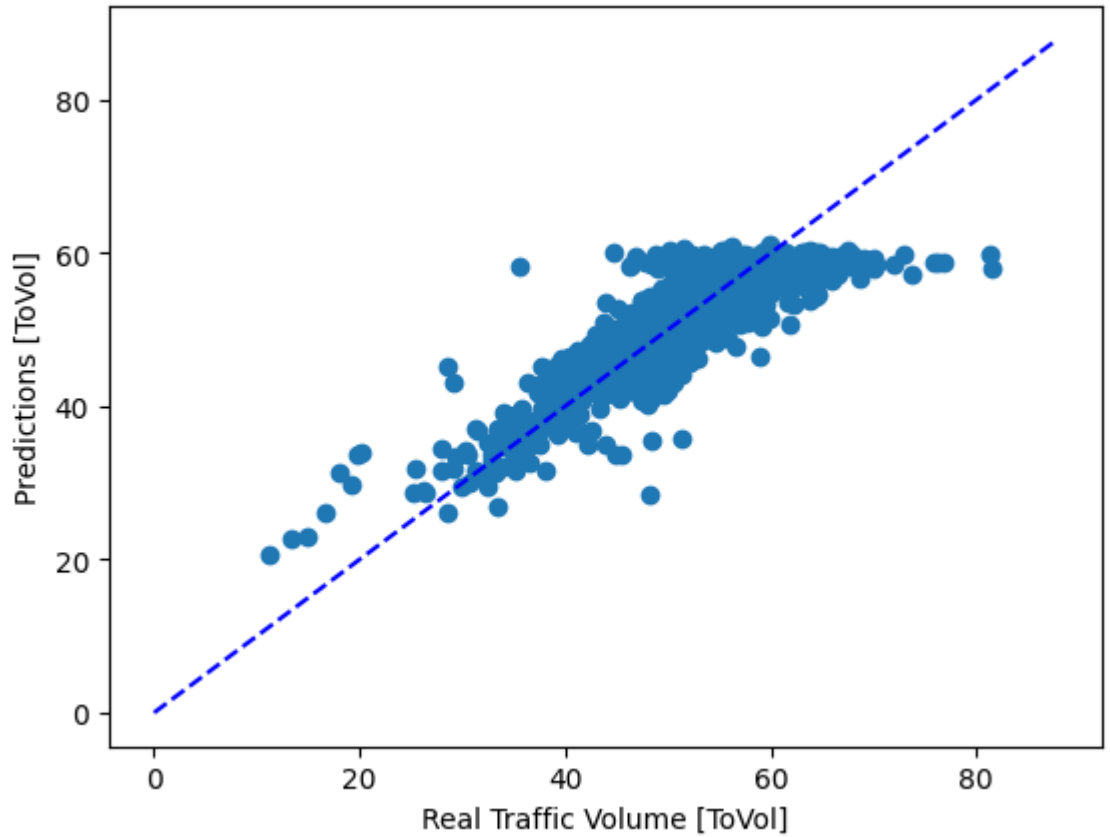
```
plt.plot(test_real[:576])
plt.plot(test_prediction[:576])
plt.xticks(rotation=45)
plt.xticks([0, 144, 288, 432,575])
plt.xlabel('Real [Speed]')
plt.ylabel('Predictions [Speed]')
```

대전시 유성구 대학로 (vds16)



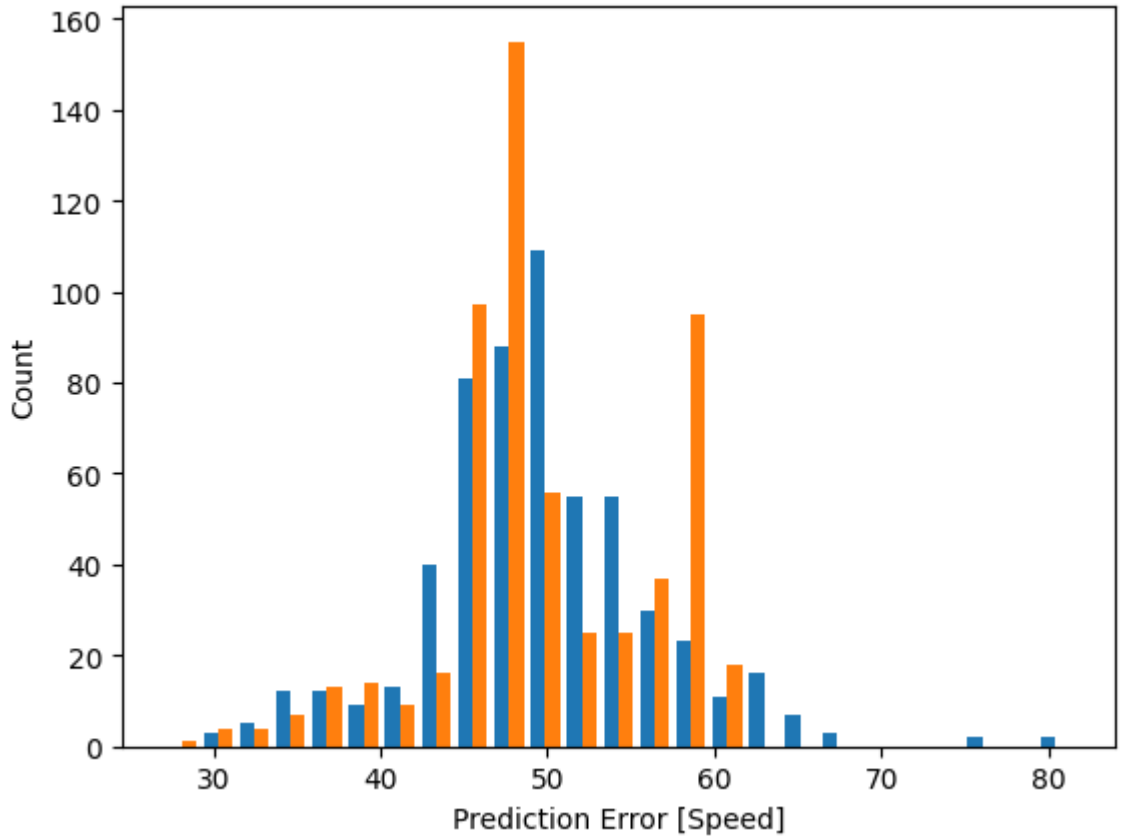
DNN 회귀 예측 오차

```
plt.scatter(test_real, test_prediction)
plt.xlabel('Real Traffic Volume [ToVol]')
plt.ylabel('Predictions [ToVol]')
plt.plot([0, max_speed], [0, max_speed],'b--')
```



회귀 예측 오차

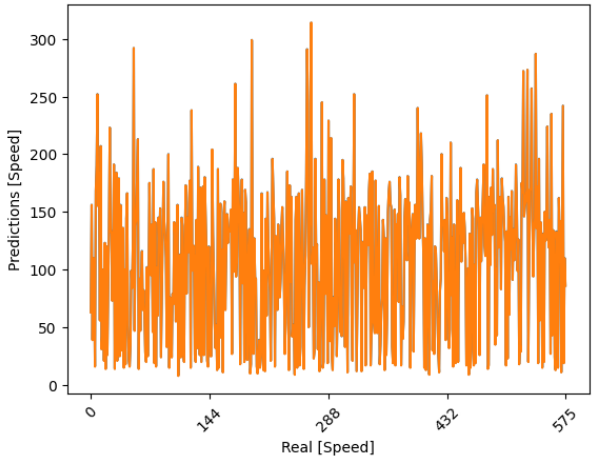
```
error = test_real[:576], test_prediction[:576]
plt.hist(error, bins = 25)
plt.xlabel("Prediction Error [Speed]")
plt.ylabel("Count")
```



프로젝트

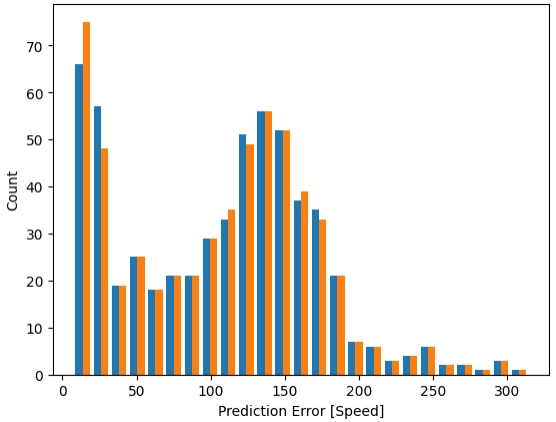
교통량 예측 : ToVol

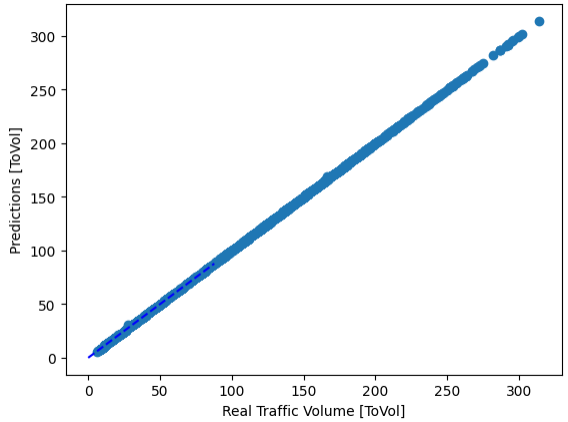
문제:
총 교통량은 작은차량 + 중형차량 + 대형차량의 교통량 합을 신경망으로 증명해 보세요.
ToVol = SmVol + MeVol + LaVol, 즉, 34 + 9 + 0 = 43



	Date	ToVol	SmVol	MeVol	LaVol	Speed	OccRate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84

```
features = ['SmVol', 'MeVol', 'LaVol']
X = df[features]
y = df.iloc[:,0:1].values   # ToVol
```

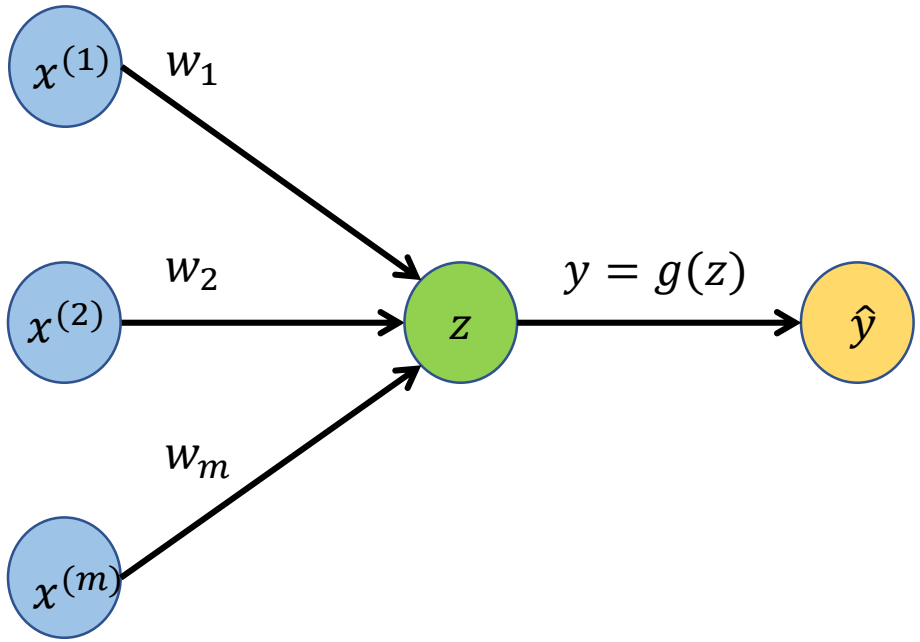




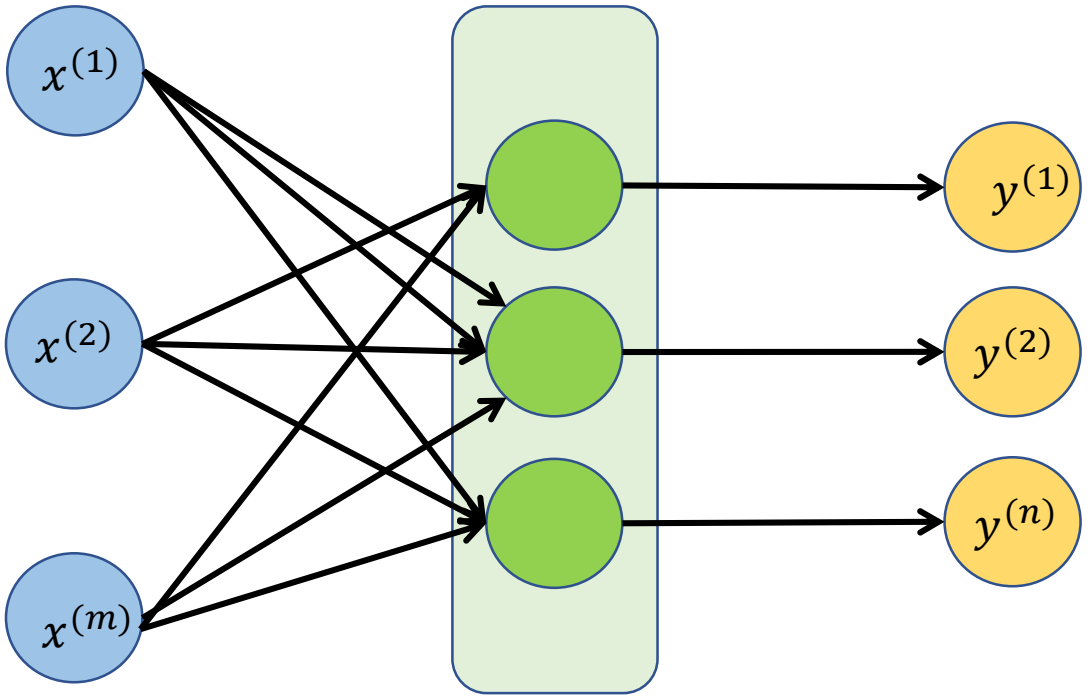
8) SimpeRNN 모델과 교통 흐름 예측하기

The Perceptron Revisited

Perceptron



Feed-Forward Networks

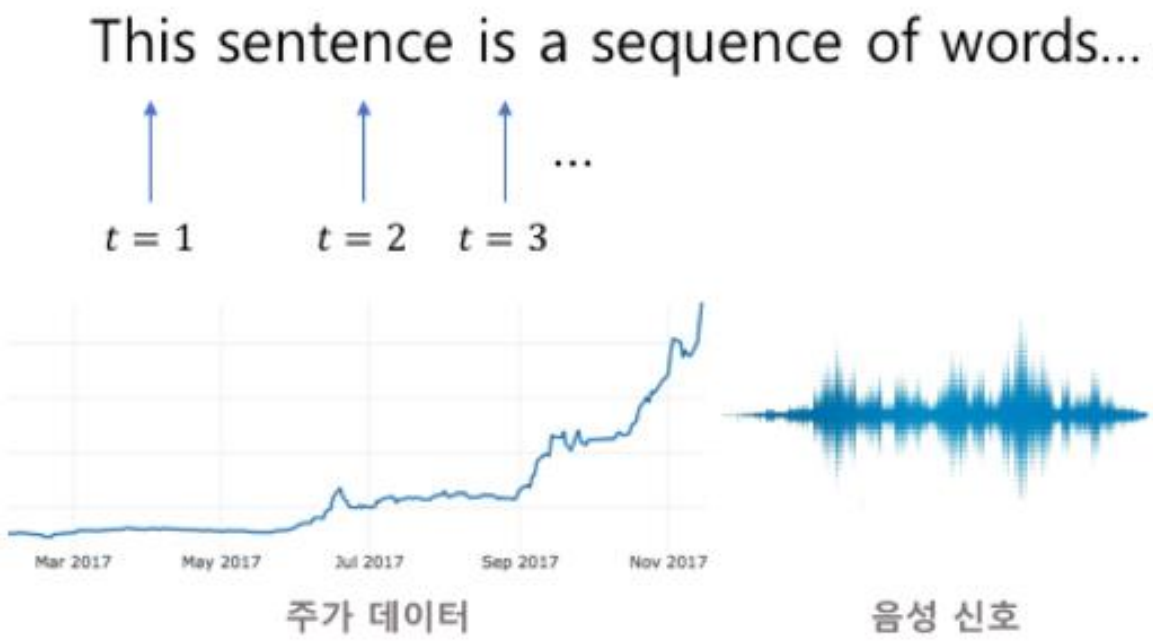


$\boldsymbol{x \in R^m}$

$\boldsymbol{\hat{y} \in R^n}$

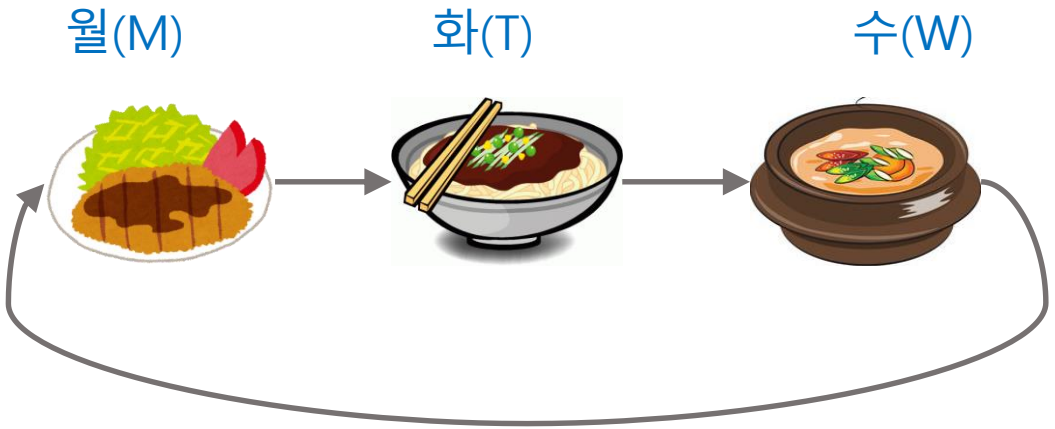
What is a time series problem? Order

- Time series deals with data over time: Predicting
  - ✓ weather (short term forecasts), climate (long term forecasts)

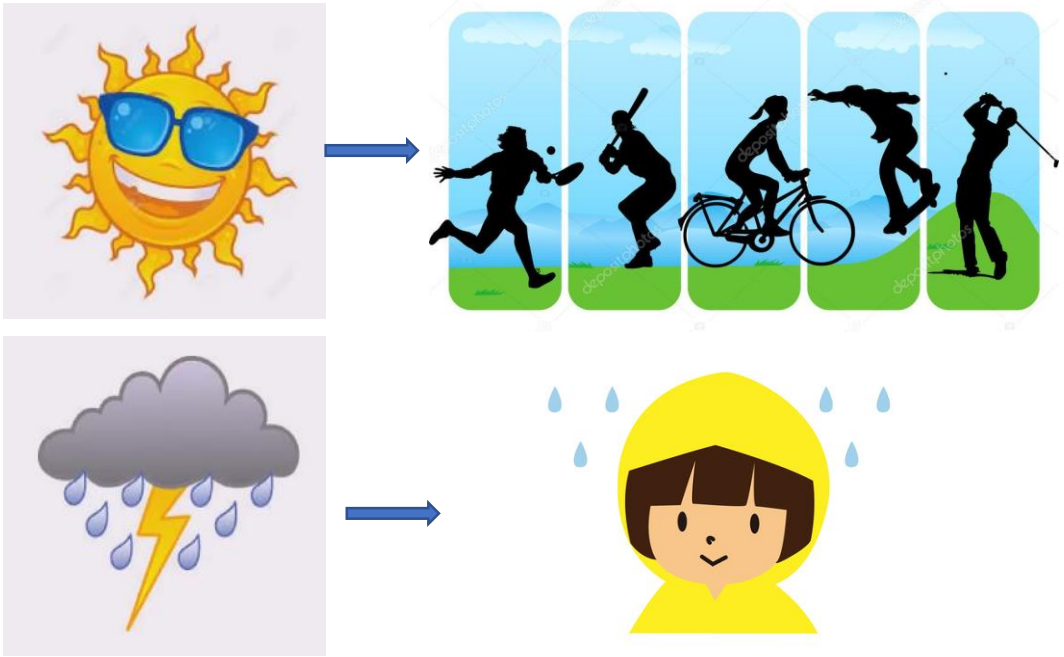


Order is important in the sequence data

3-days
cooking pattern

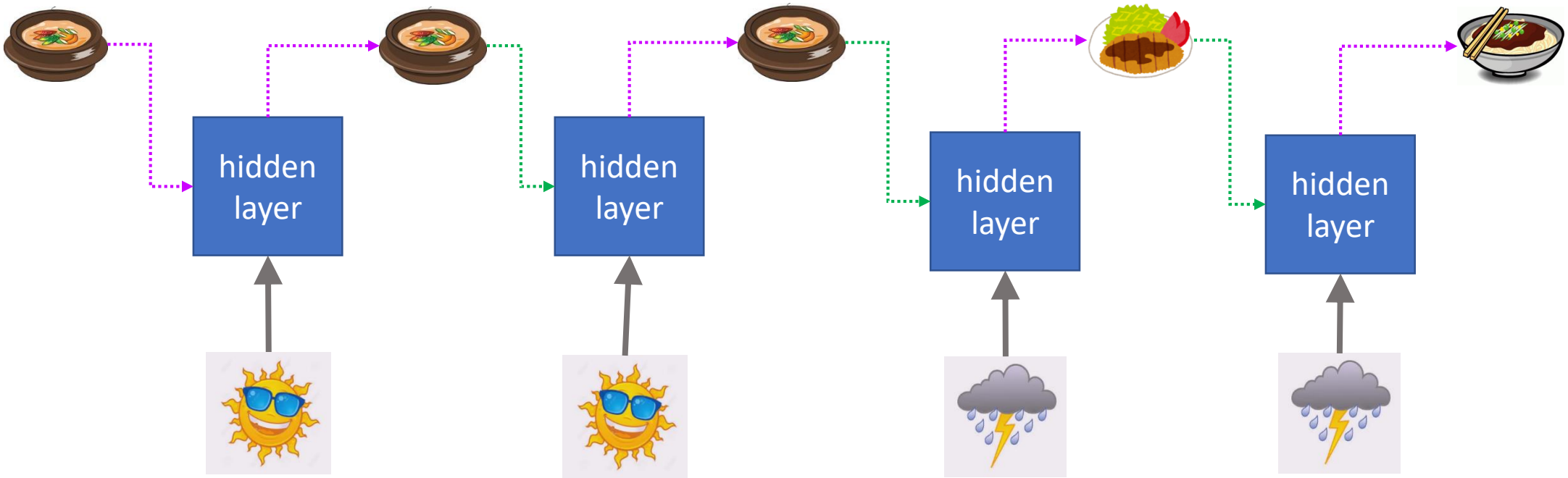


It’s sunny, the same food as yesterday
It’s rainy, following the pattern



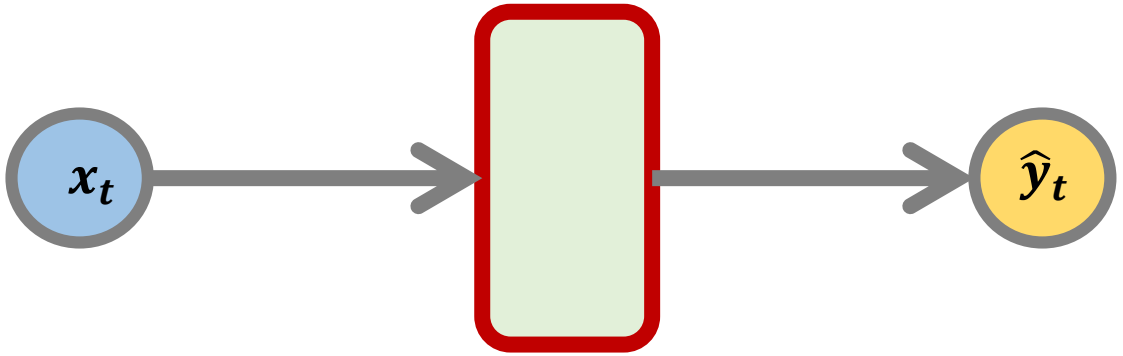
Input, Memory, and Recurrent

SimpleRNN(1, input_shape=(4,1))



Feed-Forward Networks Revisited

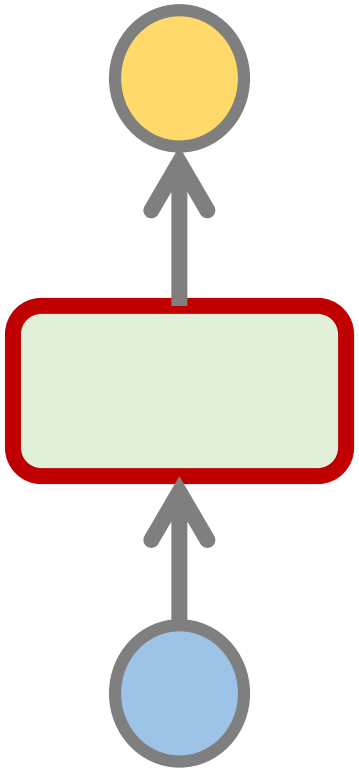
뉴런 개수를 1개로 만들고, 가중치(weights)를 1개로 표시



$x_t \in R^m$

$\hat{y} \in R^n$

90도 회전

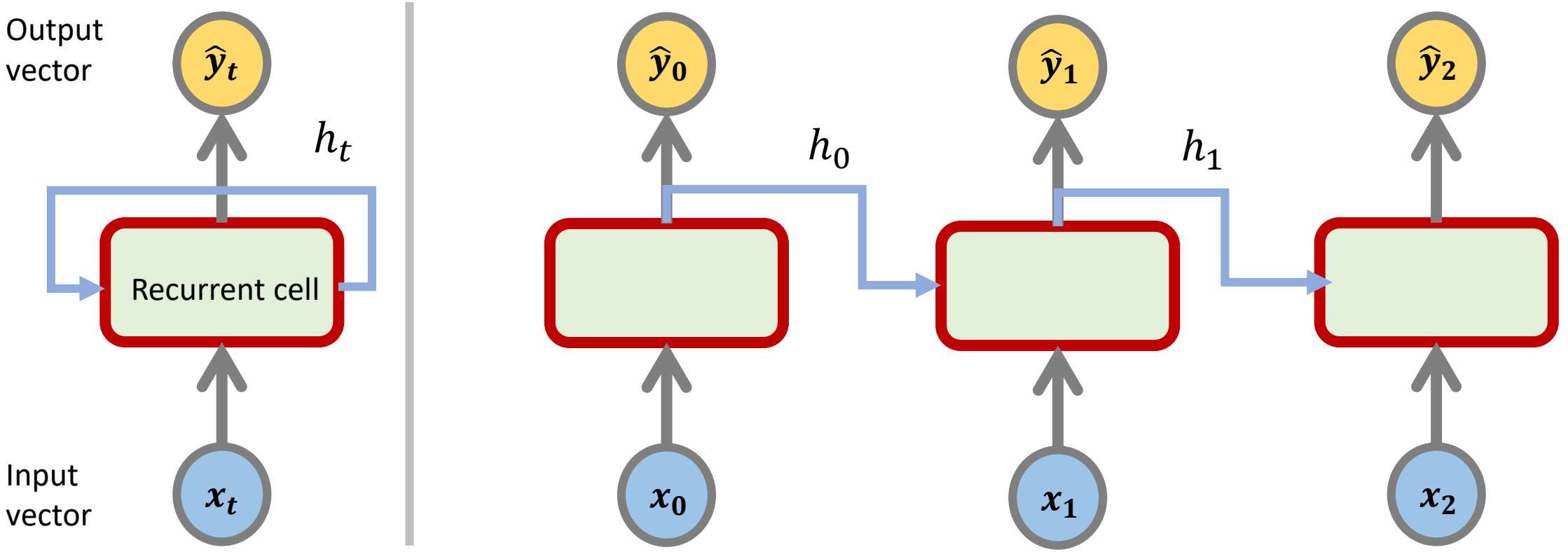


Neurons with Recurrence

$$\hat{y}_t = f(x_t, h_{t-1})$$

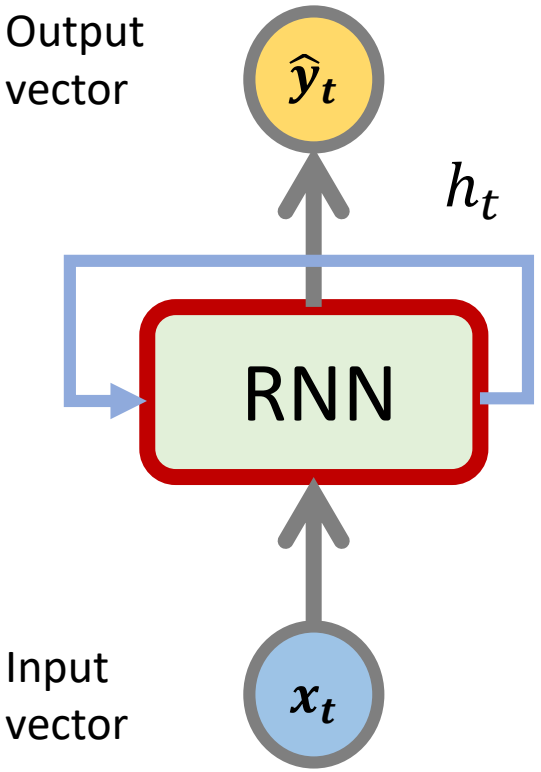
output input past memory

기억: 과거 상태 정보



Recurrent Neural Networks (RNNs)

RNNs have a state, h_t , that is updated at each time step as a sequence is processed



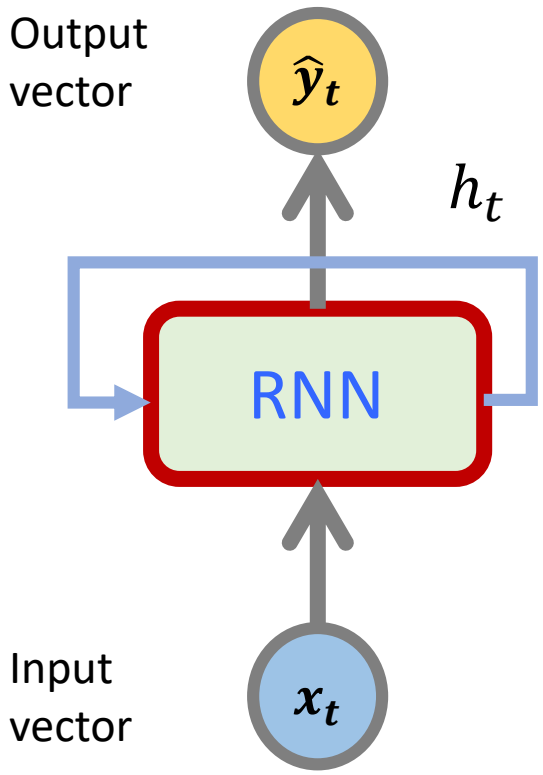
Apply a recurrence relation at every time step to process a sequence:

$$h_t = f_W(x_t, h_{t-1})$$

cell state — input — past cell state

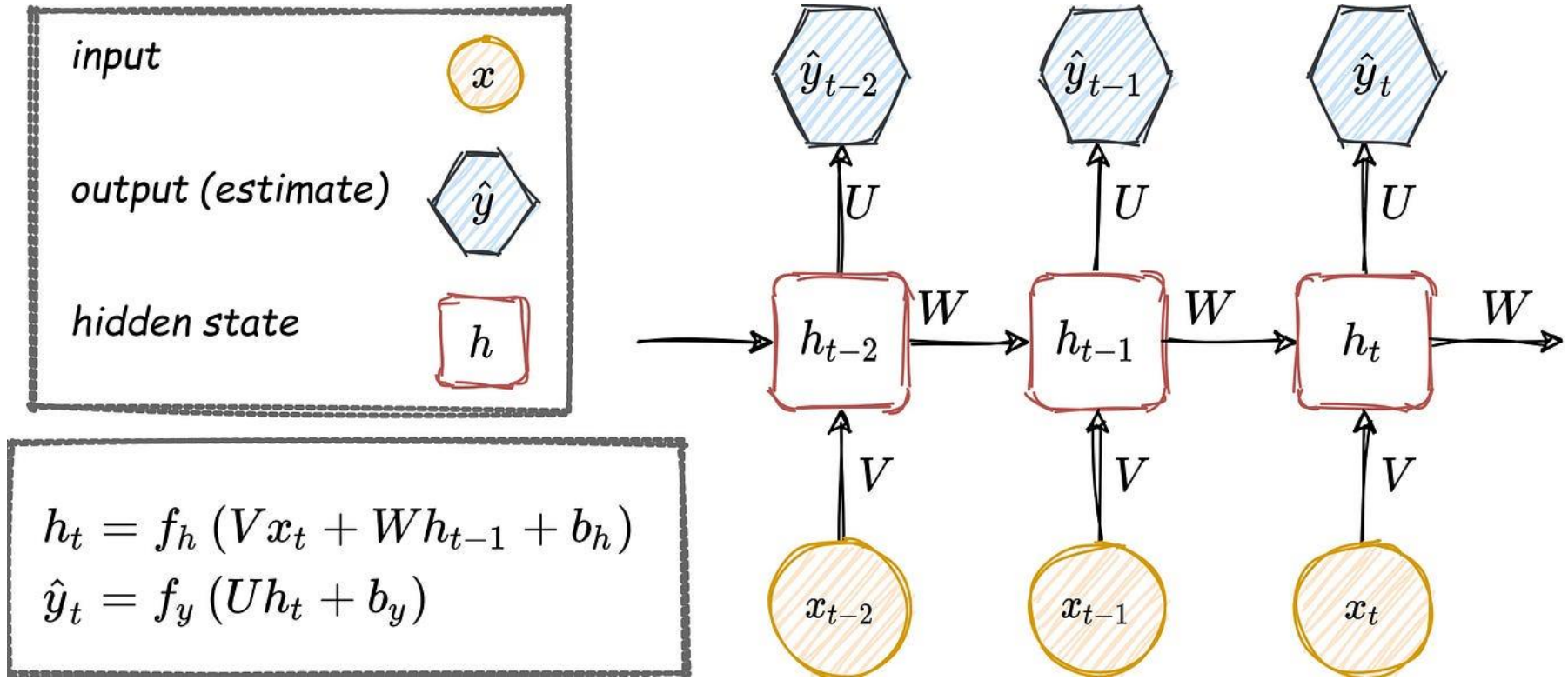
Activation function with weights W

RNN Intuition




```
model = myRNN()
hidden_state = [0,0,0,0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = model(word, hidden_state)

next_word_prediction = prediction
# >>> "networks"
```

Multy Layers Perceptrons



(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

Backpropagation Through Time (BPTT)

Loss Function $\mathcal{L}$

output (estimate) $\hat{y}$

true label y

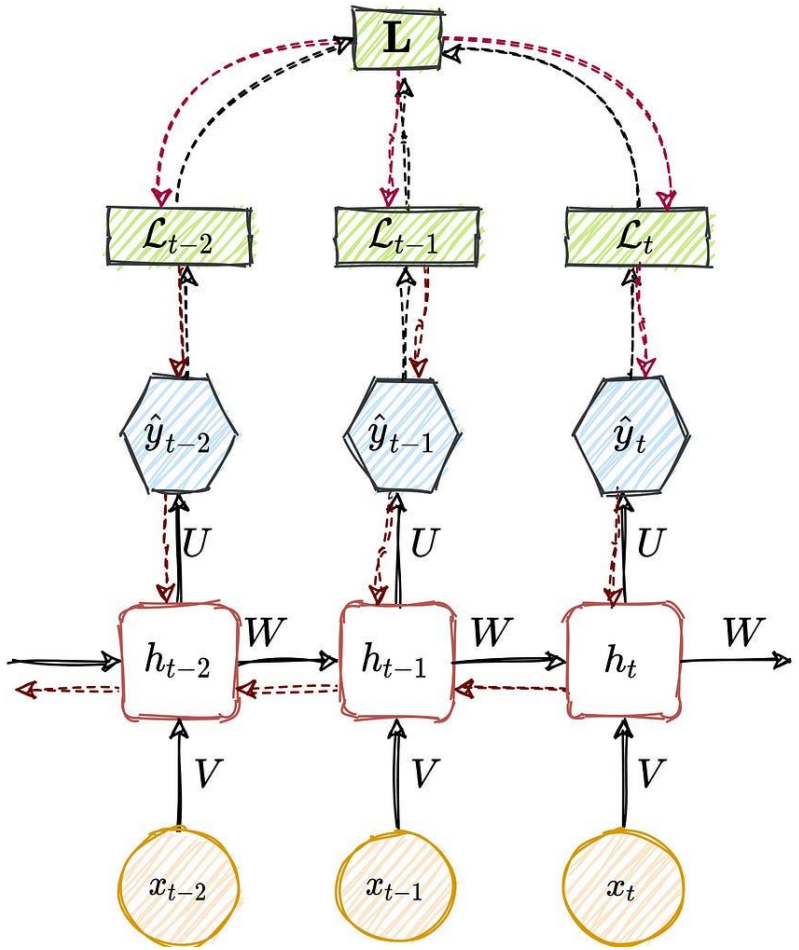
$$\mathbf{L} = \sum_i \mathcal{L}_i \left(\hat{y}_t, y_t\right)$$

Forward Pass:

$$h_t, \hat{y}_t, \mathcal{L}_t, \mathbf{L}$$

Backward Pass:

$$\frac{\partial \mathbf{L}}{\partial U}, \frac{\partial \mathbf{L}}{\partial V}, \frac{\partial \mathbf{L}}{\partial W}, \frac{\partial \mathbf{L}}{\partial b_h}, \frac{\partial \mathbf{L}}{\partial b_y}$$

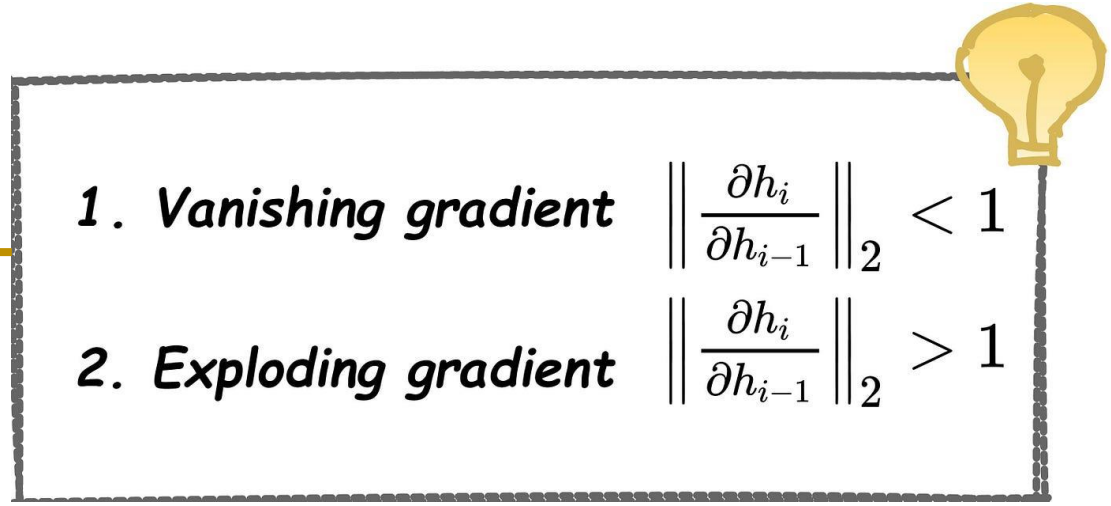


$$\frac{\partial \mathbf{L}}{\partial W} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}_i}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

$$\frac{\partial \mathbf{L}}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

The Vanishing and Exploding Gradients Problem

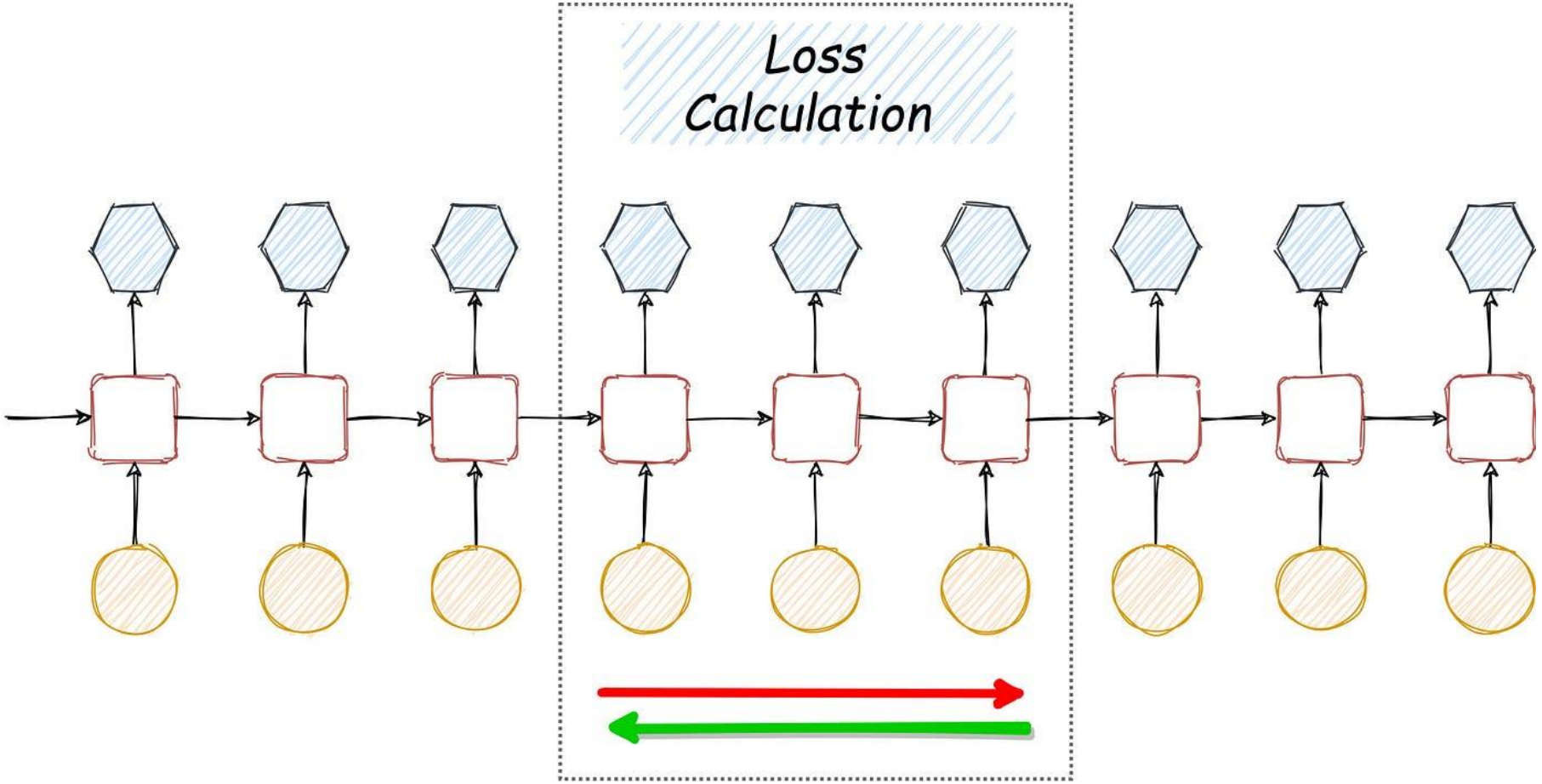


Solutions

① Truncated backpropagation through time (TBPTT)
② Long short-term memory (LSTM)
③ Gradient Clipping

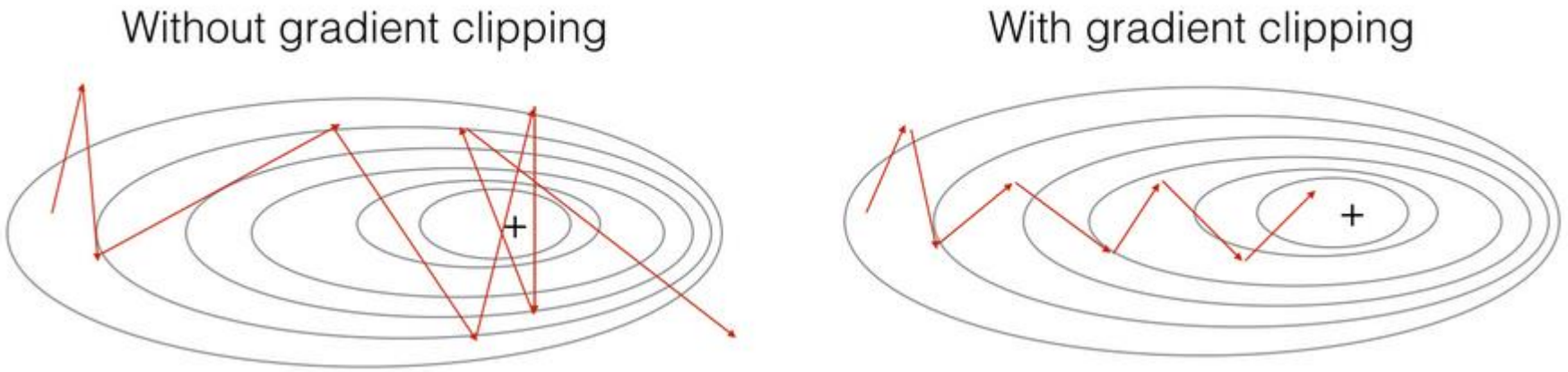
(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

Truncated Backpropagation Through Time

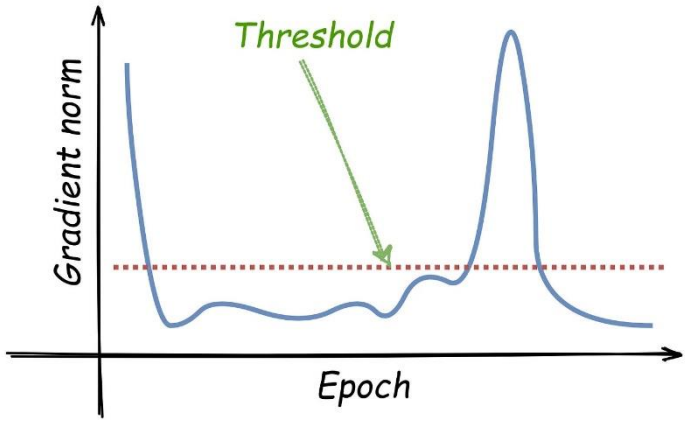


(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

Gradient Clipping



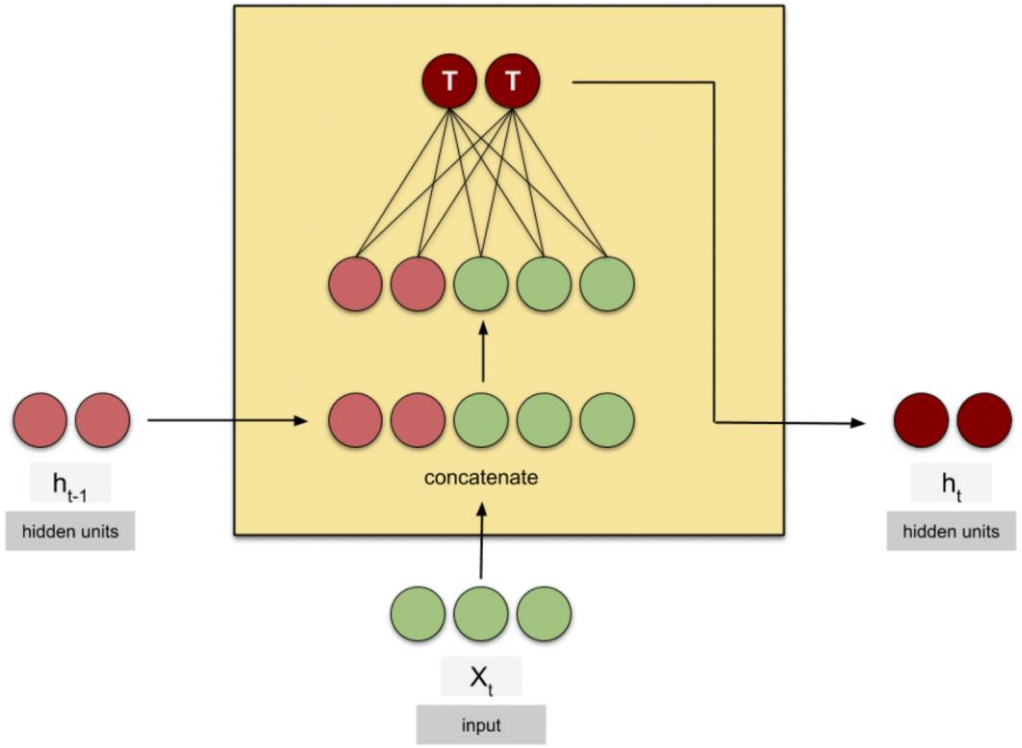
$$clip\left(g, treshold\right) = g \cdot \max\left(1, \frac{treshold}{\|g\|}\right)$$

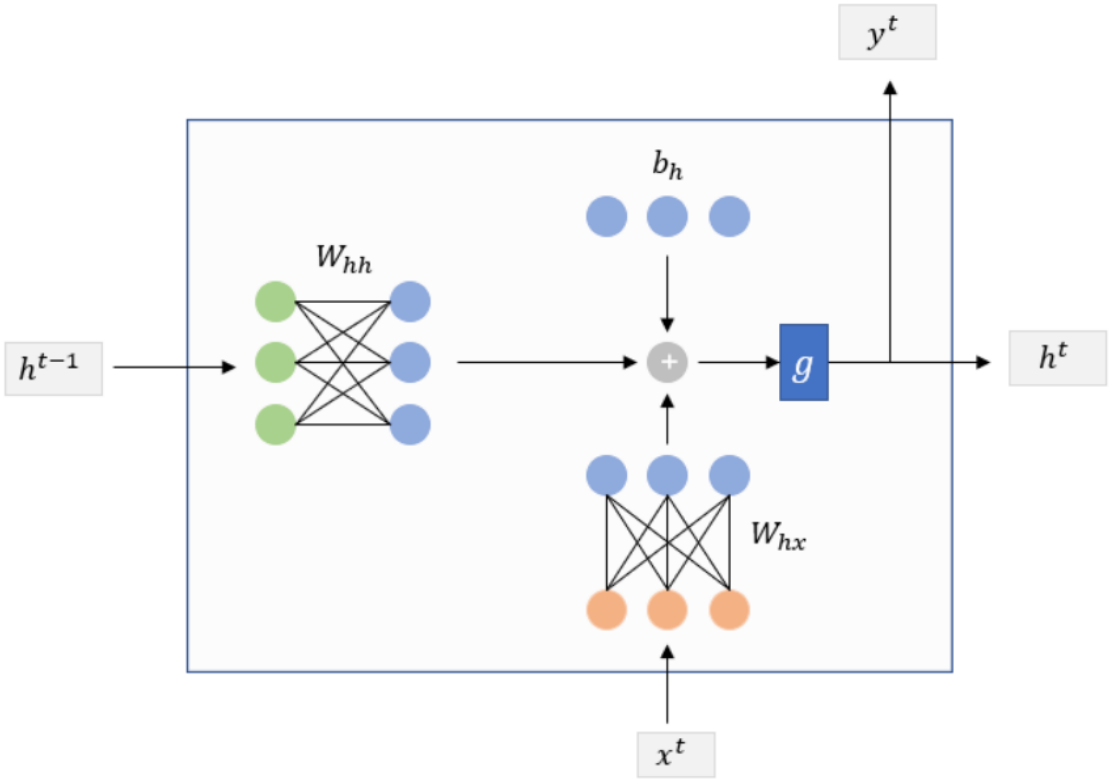


(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

9) LSTM과 GRU 모델로 교통 흐름 예측 (One-head ahead prediction)

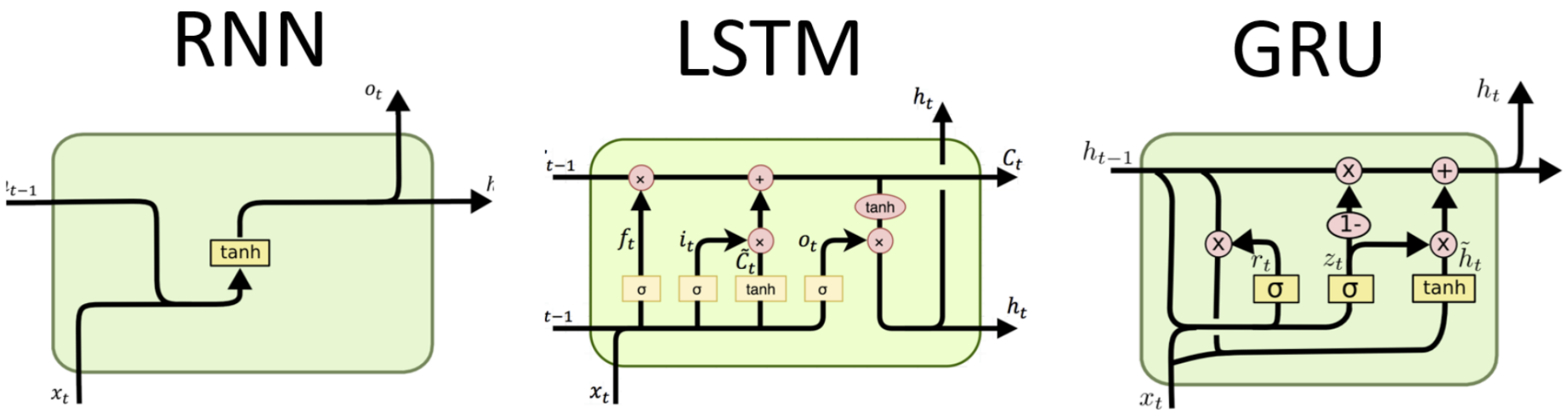
Vanilla RNN and SimpleRNN





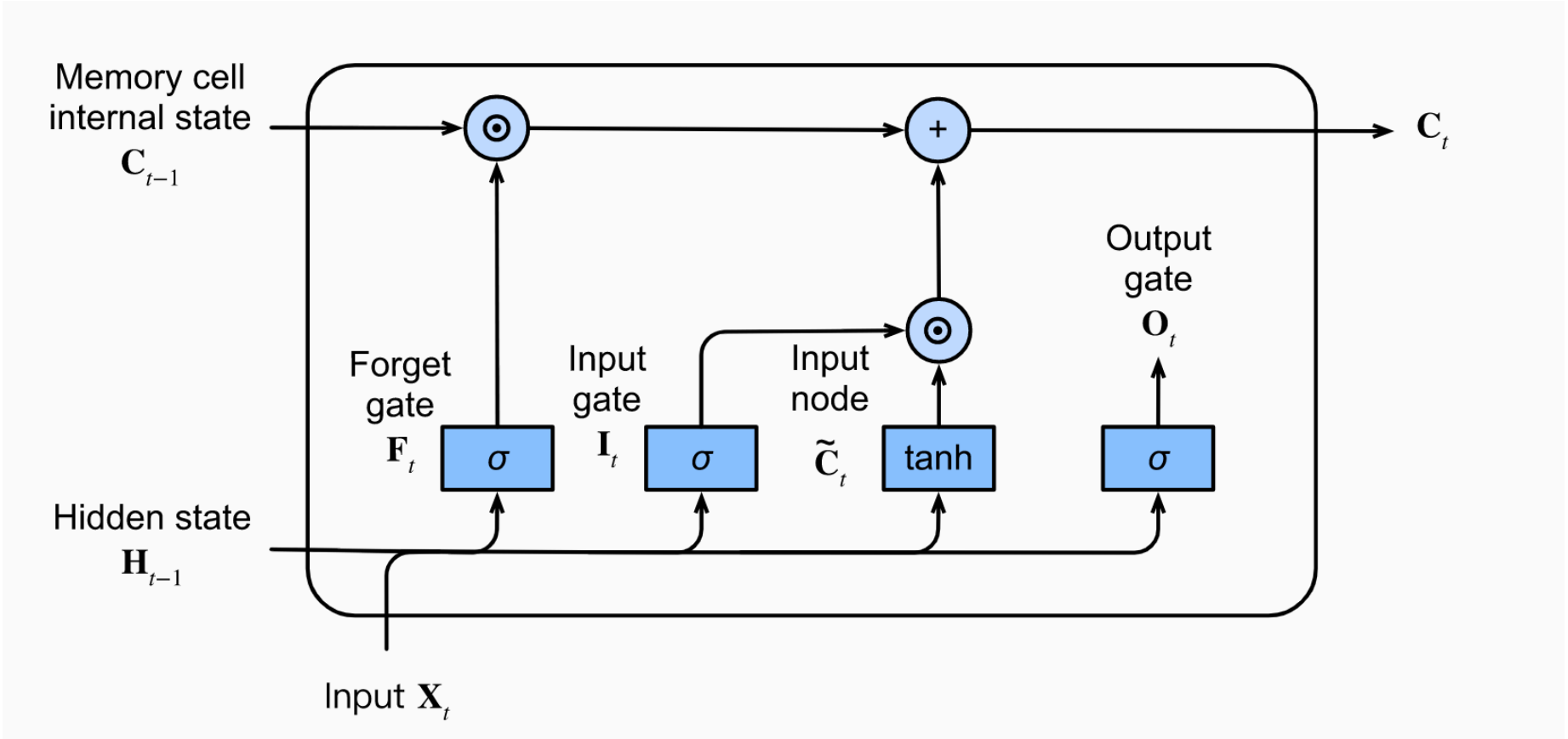
https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN

RNN, LSTM & GRU



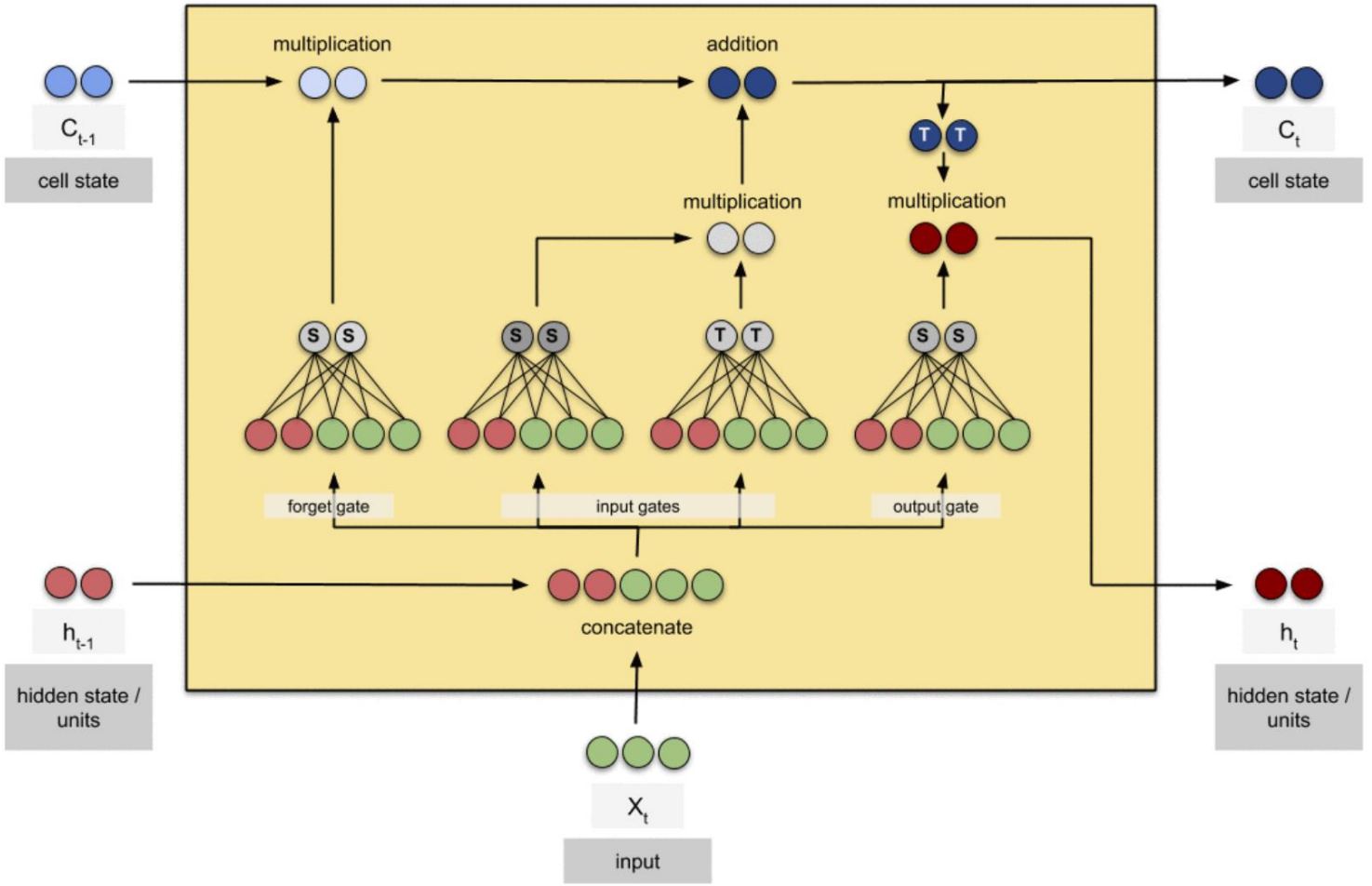
http://dprogrammer.org/rnn-lstm-gru

Long-Short Term Memory



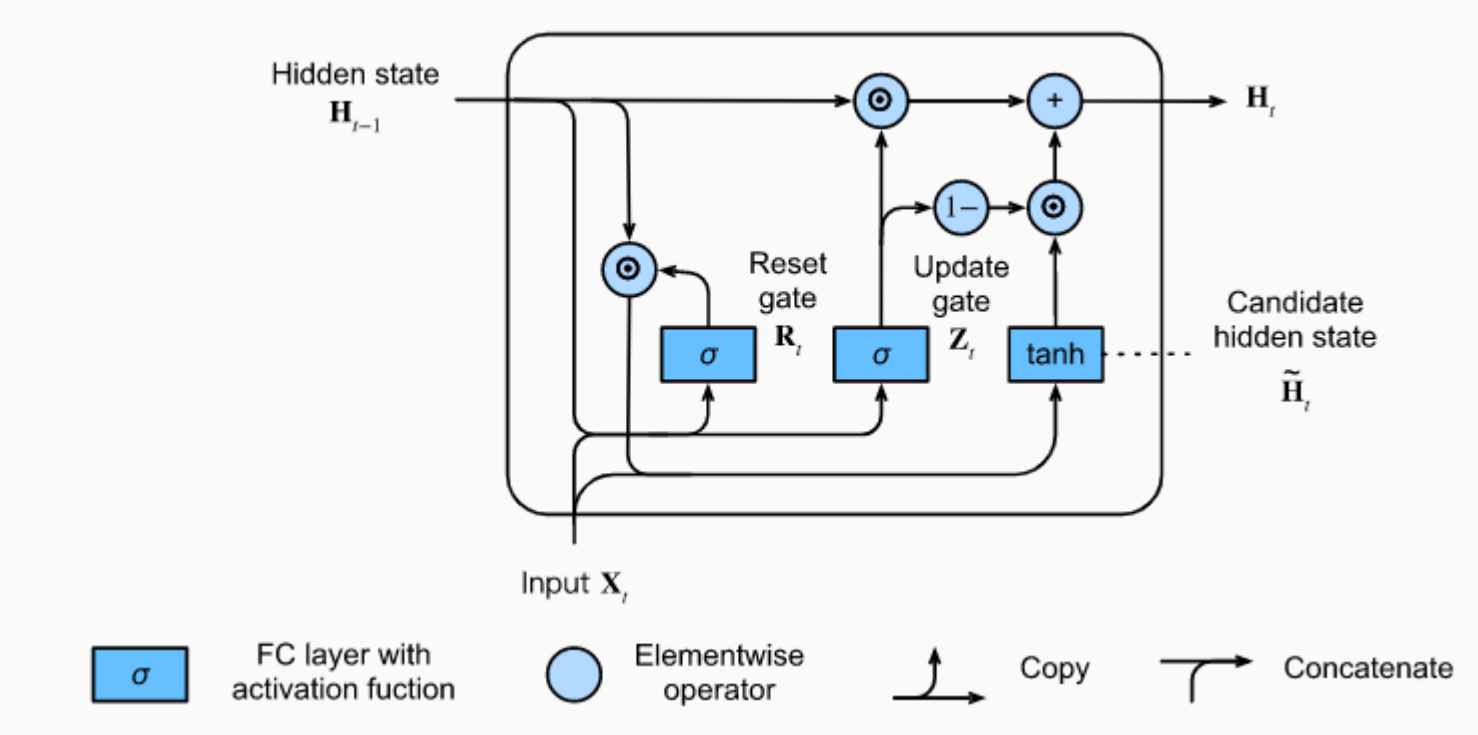
(source) https://d2l.ai/chapter_recurrent-modern/lstm.html

LSTM: Long-Short Term Memory



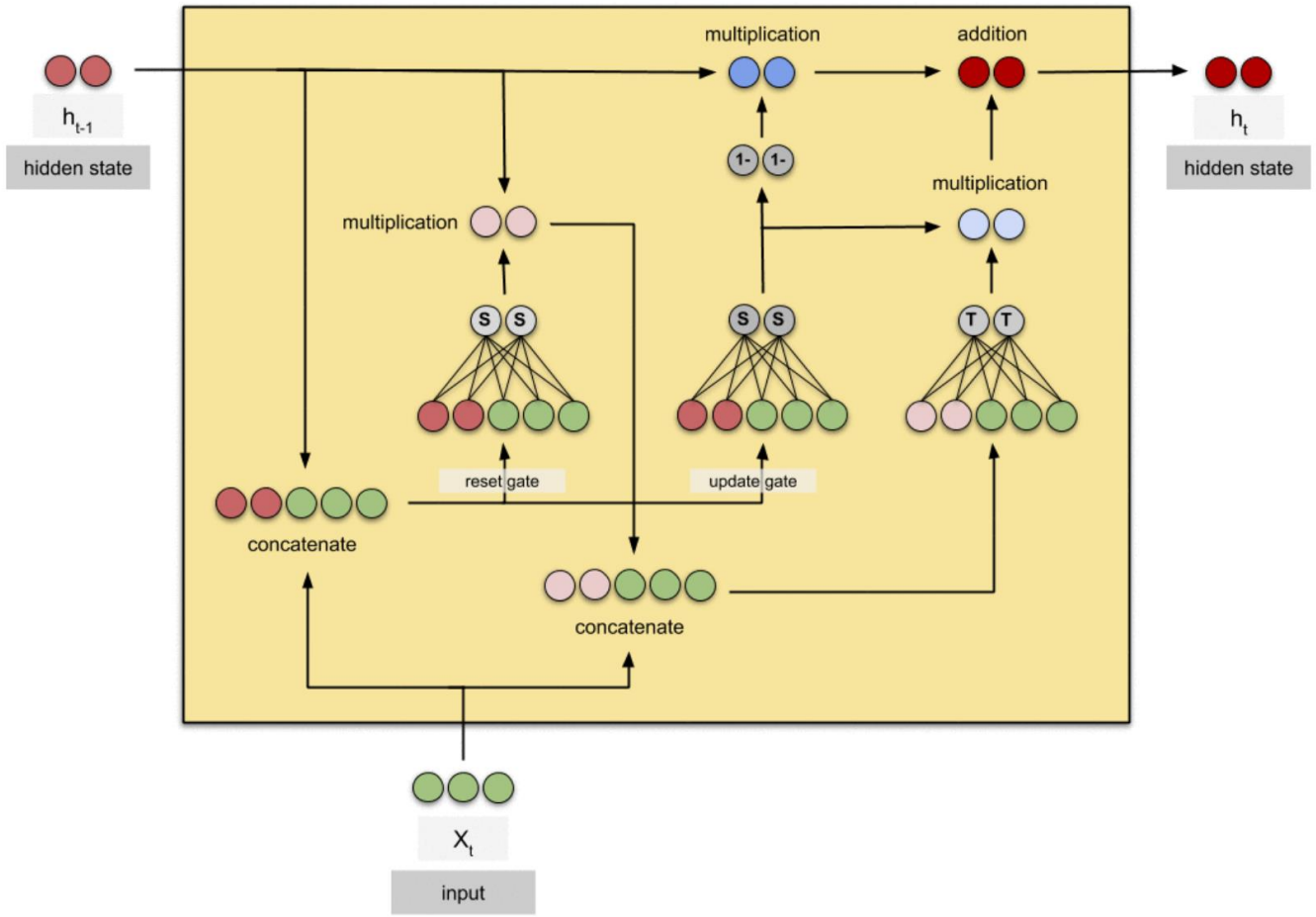
https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

Gated Recurrent Units



https://d2l.ai/chapter_recurrent-modern/gru.html

Animated GRU



https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

Number of parameters for SimpleRNN

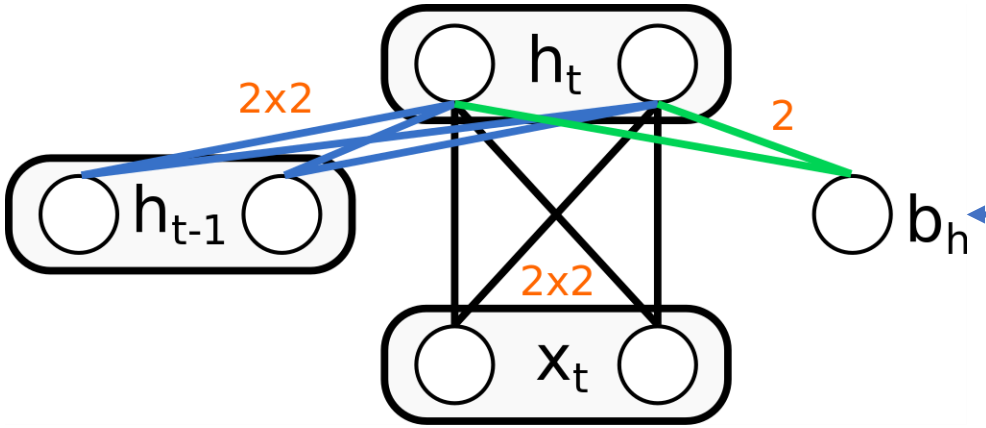
```
model = Sequential([SimpleRNN(2, input_shape=(2,2)) ])
```

Neuron=2,
Feature=2

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)       (None, 2)                 10
```

neuron*(neuron+feature+bias): 2*(2+2+1)=10

$$h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$



Number of parameters for LSTM

```
xInput = Input(batch_shape=(None, 5, 1))
xLstm = LSTM(3)(xInput)
xOutput = Dense(1)(xLstm)

model = Model(xInput, xOutput)
```

Sharing Weight for all LSTM

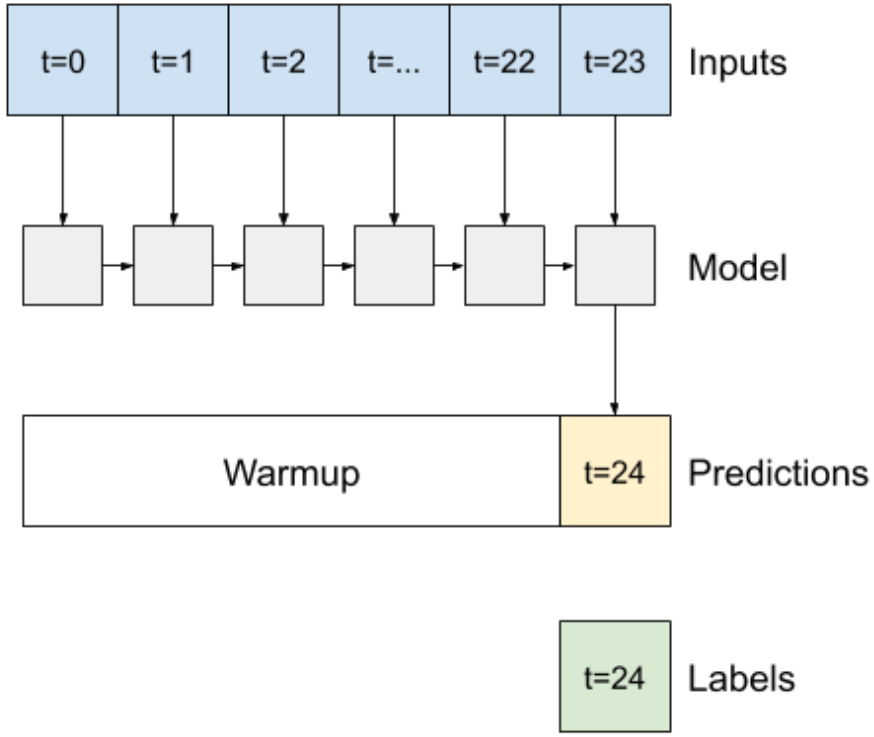
```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 5, 1)]            0
lstm_6 (LSTM)                (None, 3)                 60
dense_6 (Dense)              (None, 1)                 4
=================================================================
Total params: 64
Trainable params: 64
Non-trainable params: 0
```

# of Para $= 4 * (n + m + 1) * m$

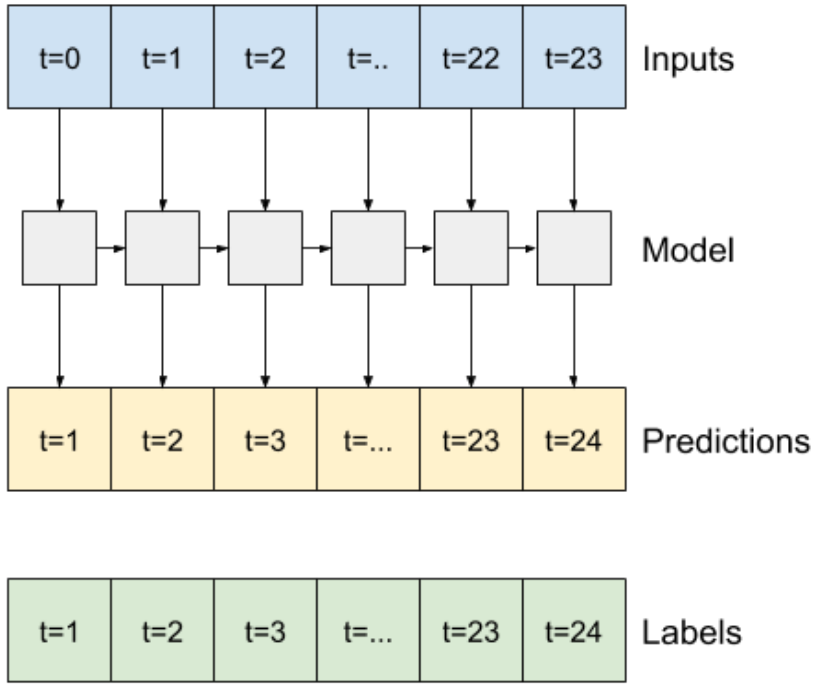
Here, n=1, m=3,
Total parameters = 4*(1+3+1)*3=4*5*3=20*3=60

Return_sequences and return_states

Return_sequenes=False

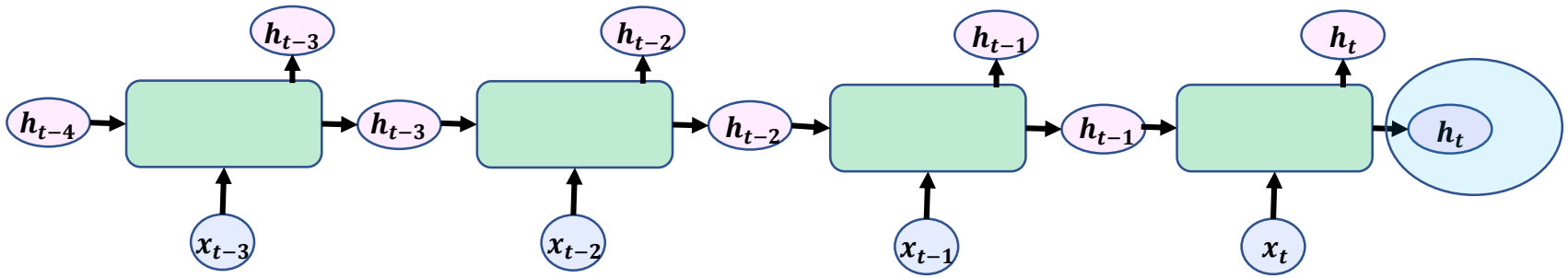


Return_sequenes=True

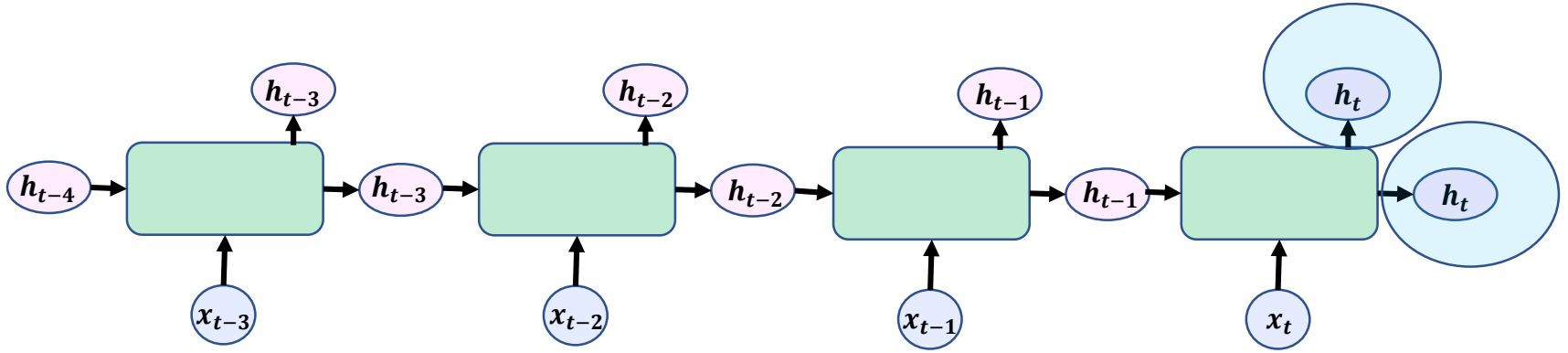


https://www.tensorflow.org/tutorials/structured_data/time_series?hl=ko

1) return_sequences=False, return_states=False

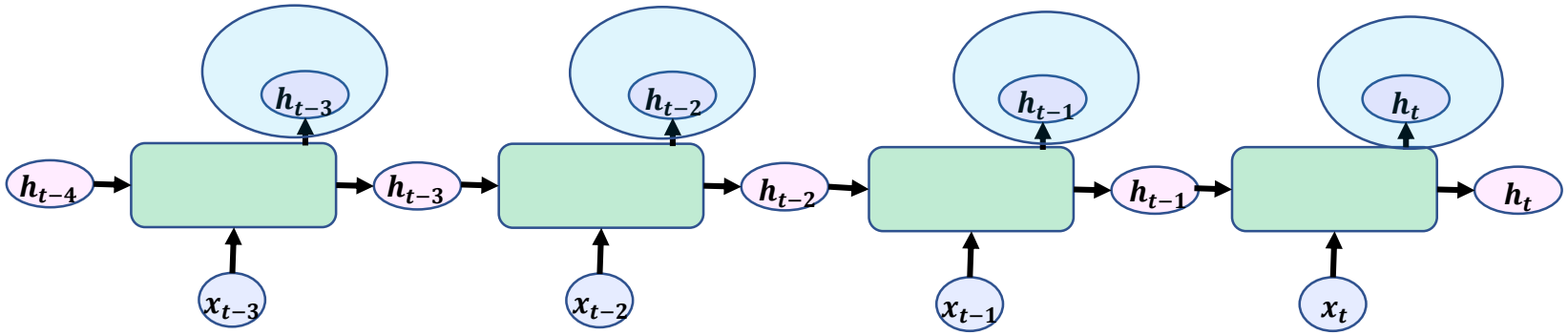


2) return_sequences=False, return_states=True

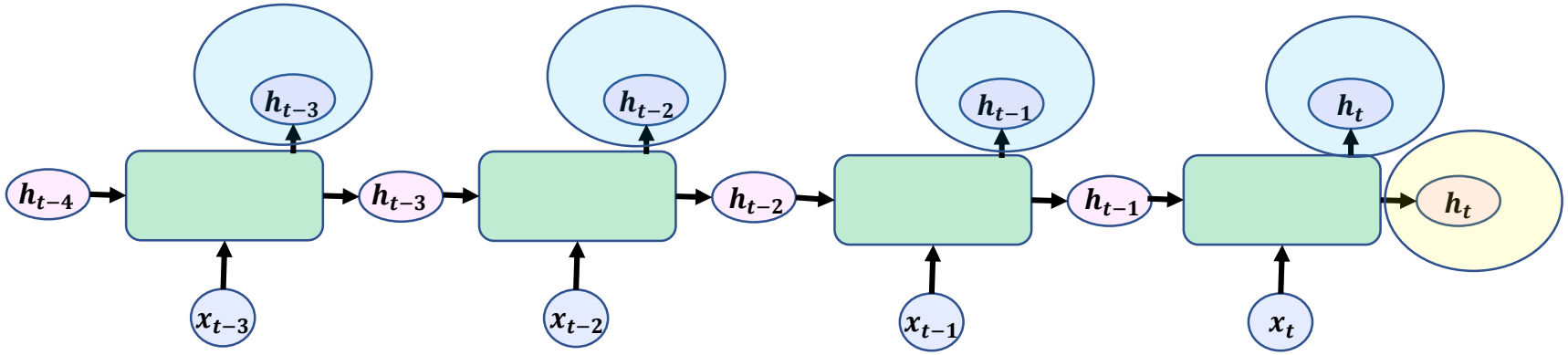


Return_sequenes=True, return_state=True

3) return_sequences=True, return_states=False



4) return_sequences=True, return_states=True



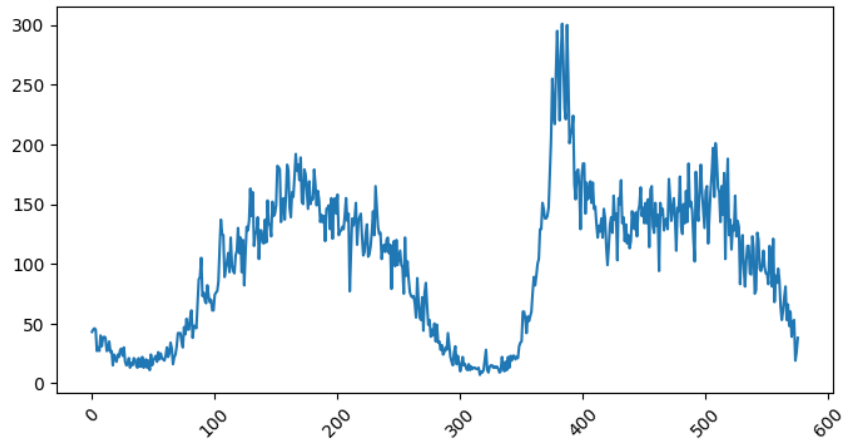
교통흐름 예측 모델 (I)
One-step ahead prediction

필요한 라이브러리 추가

```
def plot_windows(X, Y, Y_pred):
    plt.figure(figsize=(8, 4))
    plt.plot(X[0,:], "o-")
    if Y is not None:
        plt.plot(np.arange(look_back, look_back + look_forward), Y[0, :], "go", label="Actual")
    if Y_pred is not None:
        plt.plot(np.arange(look_back, look_back + look_forward), Y_pred[0, :], "rx-", label="Forecast", markersize=10)
    plt.legend(fontsize=12)
    plt.grid(True)
    plt.axvline(look_back, color='gray',linestyle='--',linewidth=2)
    plt.show()
```

```
def plot_pred():
    plt.figure(figsize=(5, 4))
    plt.scatter(y_test.flatten(), y_pred.flatten())
    plt.xlabel('Acutal')
    plt.ylabel('Predictions [Sine Wave]')
    plt.plot([0,1], [0,1],color='gray',linestyle='--',linewidth=2)
    plt.show()
```

총 교통량(ToVol)을 예측해보자




```
look_back = 12*5
look_forward = 1
```

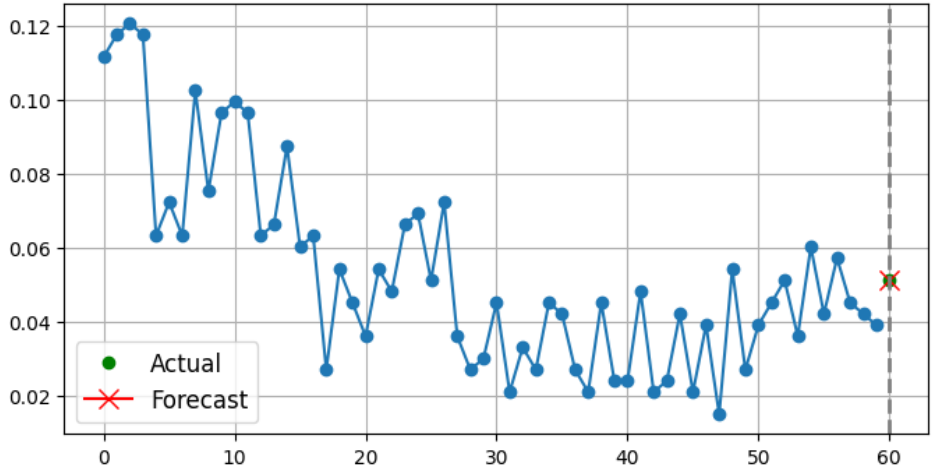
```
X, y = create_dataset(df_scaled,look_back,look_forward)
```

```
X.shape, y.shape
```

```
((8003, 60, 1), (8003, 1, 1))
```

```
plot_windows(X,y,y)
```

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back, look_forward):
    dataX, dataY = [], []
    np.array(dataY)
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back)]
        if len(dataset[i + look_back:i+look_back+look_forward]) == look_forward:
            dataX.append(a)
            dataY.append(dataset[i + look_back:i+look_back+look_forward])
    return np.array(dataX), np.array(dataY)
```



1) many-to-one with Baselines (MSE)

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, shuffle=False)
print('X_train:', X_train.shape)
print('X_test :', X_test.shape)
print('y_train:', y_train.shape)
print('y_test :', y_test.shape)
```

```
X_train: (6402, 60, 1)
X_test : (1601, 60, 1)
y_train: (6402, 1, 1)
y_test : (1601, 1, 1)
```

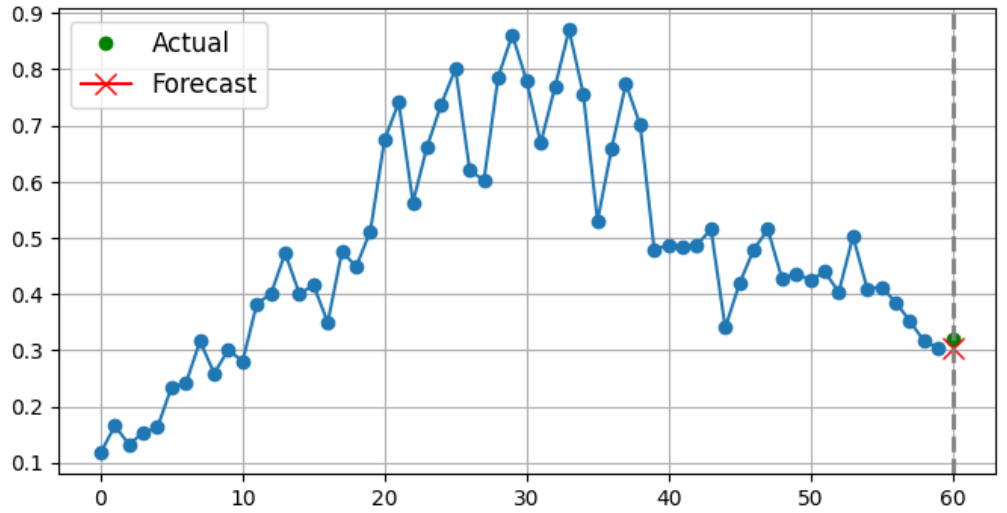
```
y_pred = X_test[:, -1]
```

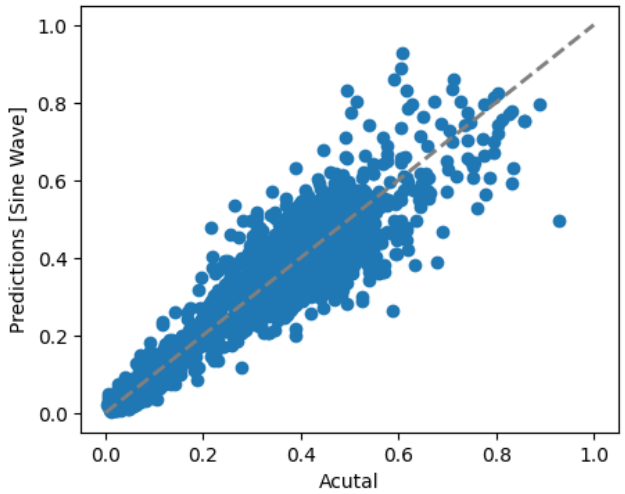
```
y_pred = y_pred[:,np.newaxis]
X_test.shape, y_test.shape, y_pred.shape
```

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

```
def model_mse():
    model = np.mean(tf.keras.losses.mean_squared_error(y_test, y_pred))
    return model
```

```
plot_windows(X_test[:, :], y_test[:,], y_pred[:,])
```

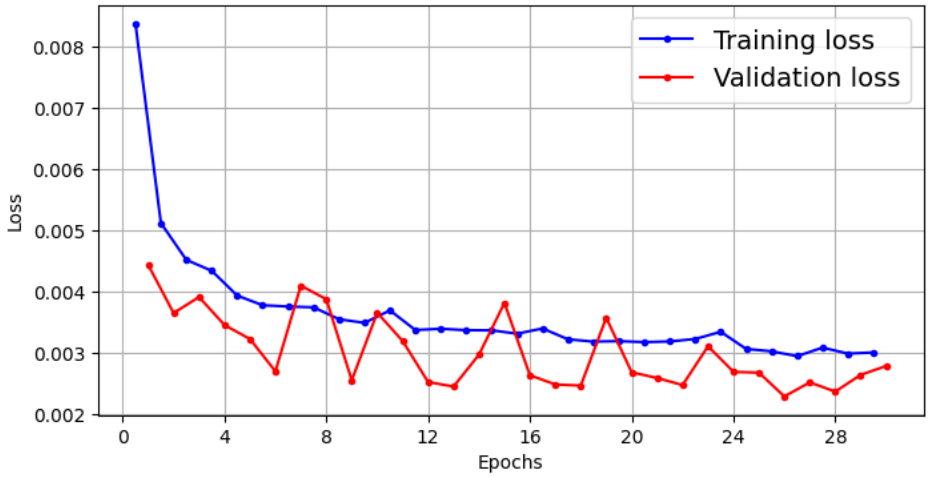





```
y_max =y_test.flatten()[:,].max()
y_min =y_test.flatten()[:,].min()
print(y_max,y_min)
plot_pred()
```

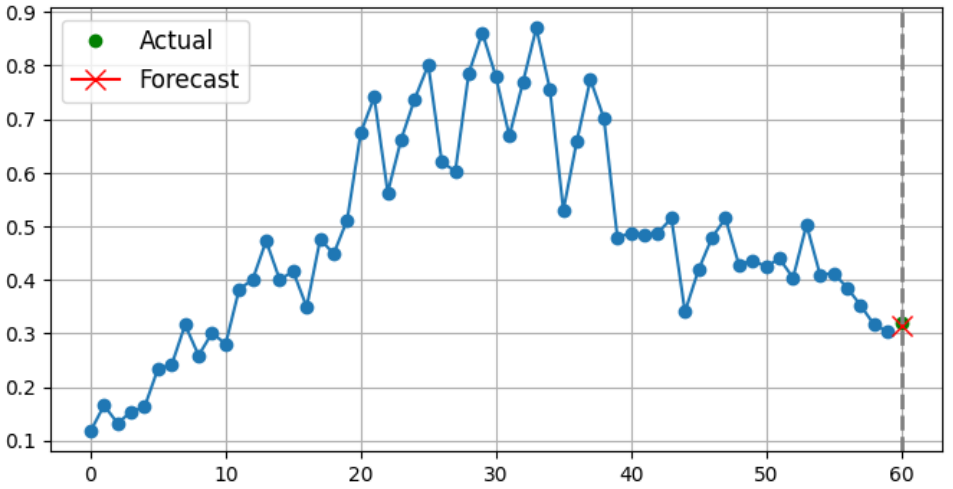
2) many-to-one with DNN

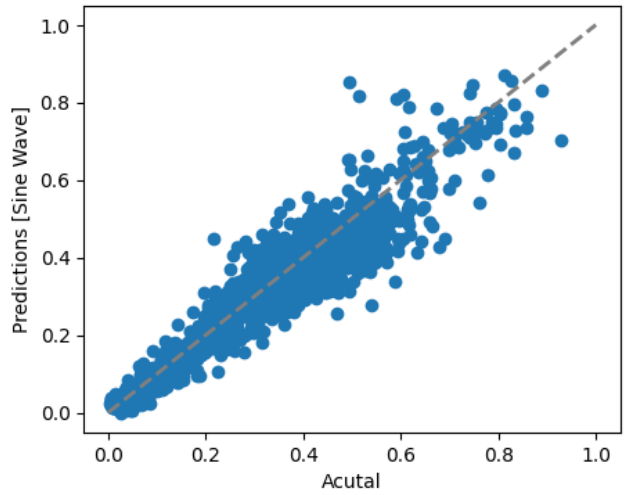
```
def model_dnn():
    model = Sequential([ Flatten(input_shape=[look_back,1]),
                         Dense(32, activation='relu'),
                         Dense(32, activation='relu'),
                         Dense(look_forward) ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```




```
a2, _ = model.evaluate(X_test, y_test)
print('m2_dense: ', a2)
plot_pred()
```

```
51/51 [==============================] - 0s 2ms/step - loss: 0.0037 - mae: 0.0446
m2_dense:  0.0037173747550696135
```

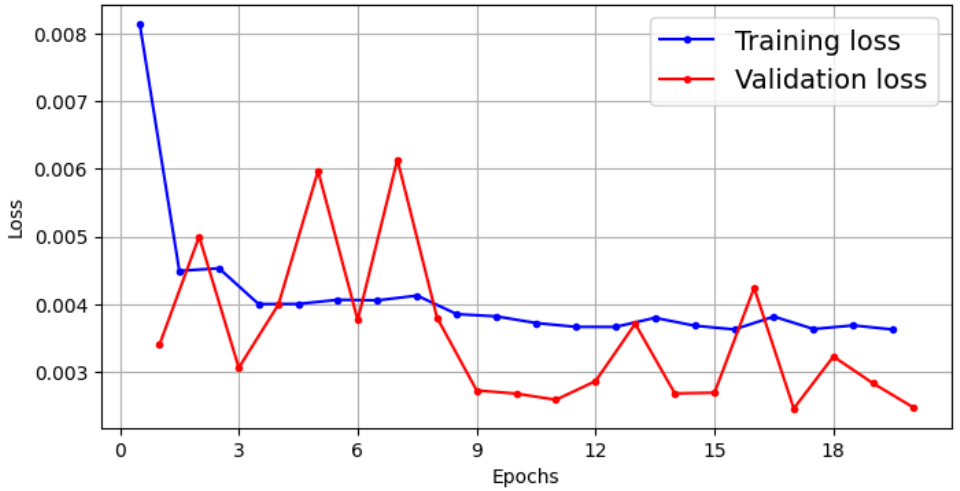


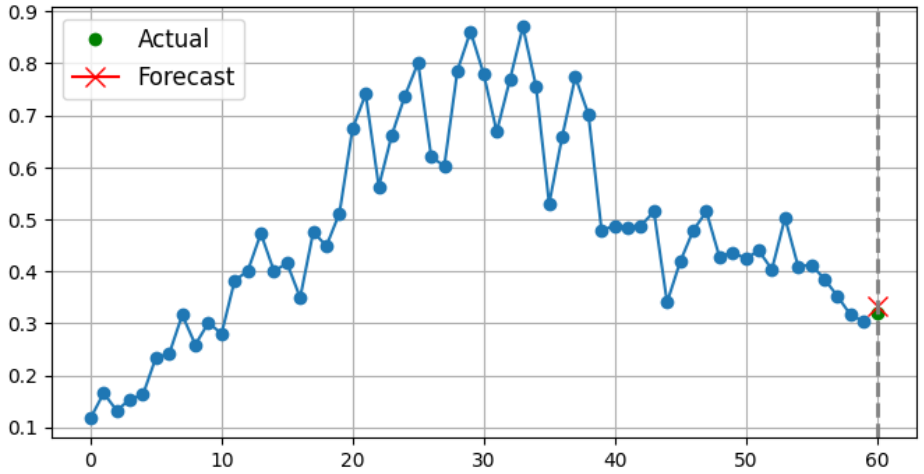


3) many-to-one with SimpleRNN

```
def model_simpleRNN():
    model = Sequential([
        SimpleRNN(32, return_sequences=True, input_shape=[look_back, 1]),
        SimpleRNN(32,return_sequences=True),
        SimpleRNN(look_forward,return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 60, 32)            1088

 simple_rnn_1 (SimpleRNN)    (None, 60, 32)            2080

 simple_rnn_2 (SimpleRNN)    (None, 1)                 34

=================================================================
```

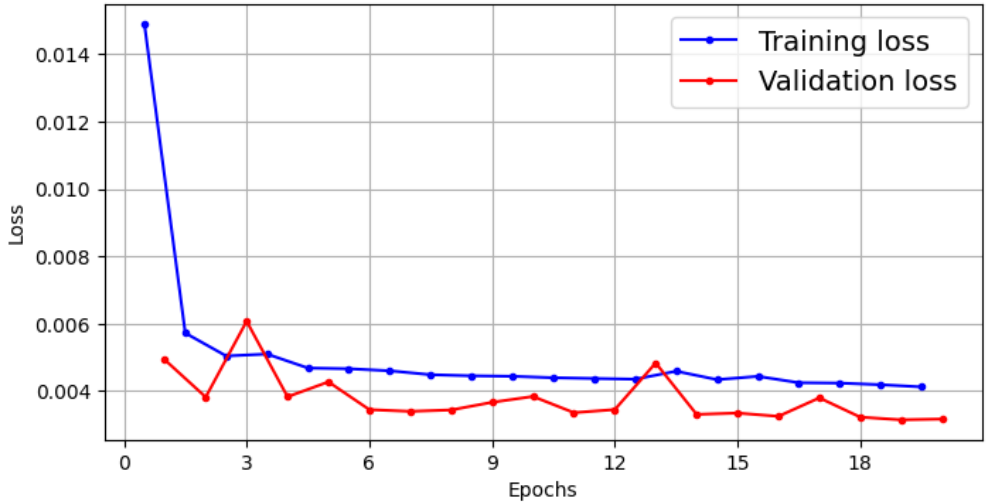




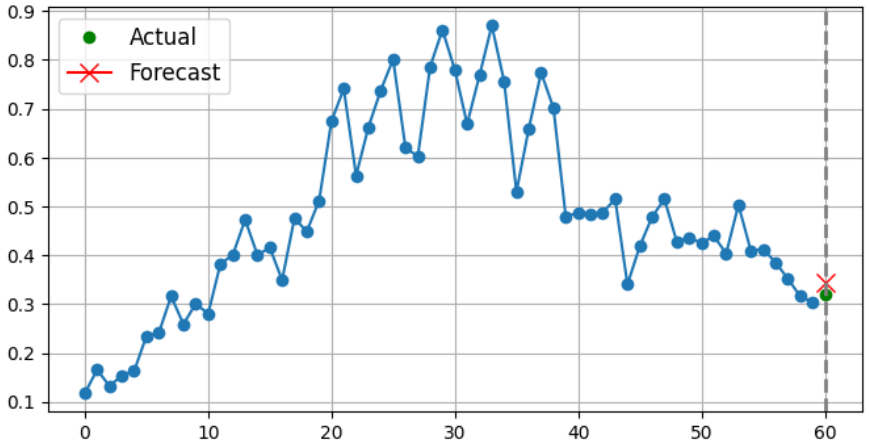

```
51/51 [==============================]
m3_simpleRnn:  0.0033664896618574858
```

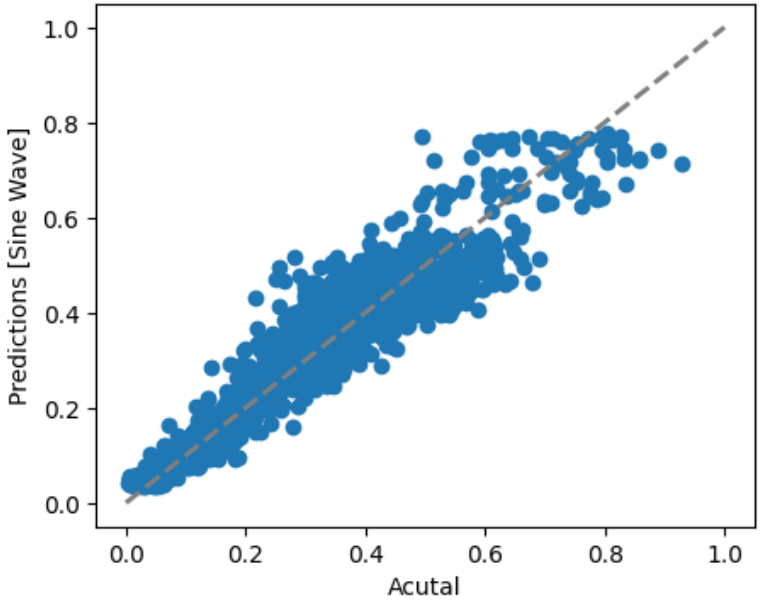
4) many-to-one with LSTM

```
def model_lstm():
    model = Sequential([
        LSTM(32, return_sequences=True, input_shape=[look_back, 1]),
        LSTM(32, return_sequences=True),
        LSTM(look_forward, return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```



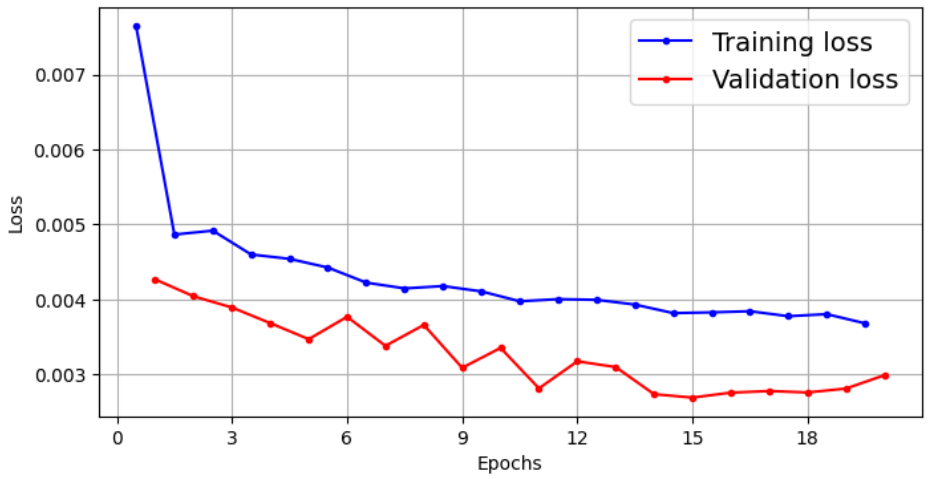

```
51/51 [==============================]
m1_LSTM:  0.003614206099882722
```



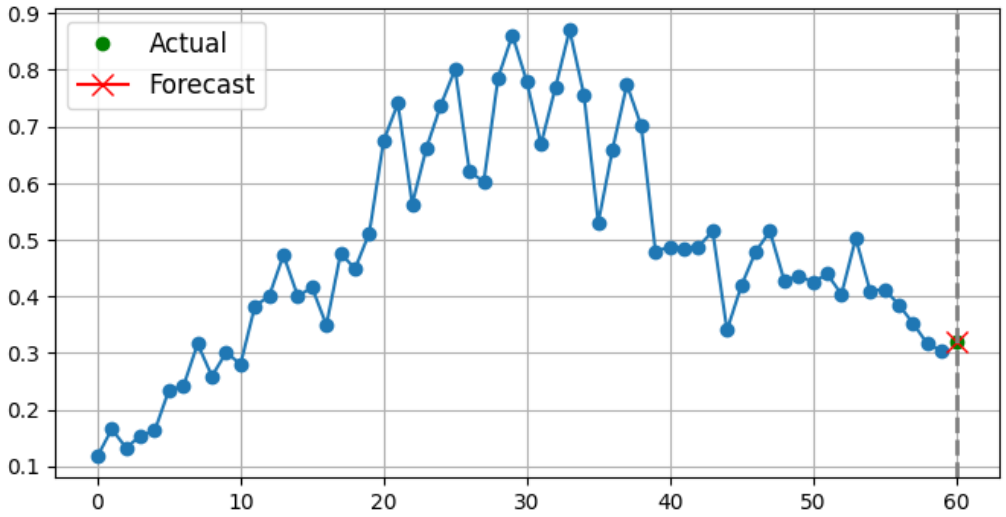


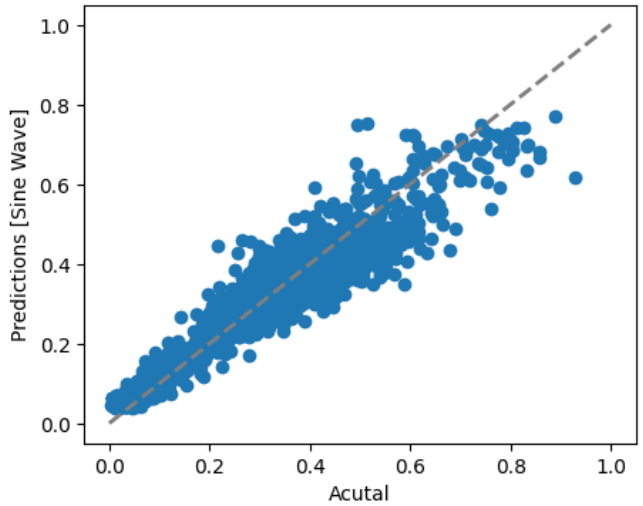
5) many-to-one with GRU

```
def model_gru():
    model = Sequential([
        GRU(32, return_sequences=True, input_shape=[look_back, 1]),
        GRU(32, return_sequences=True),
        GRU(look_forward, return_sequences=False),
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```



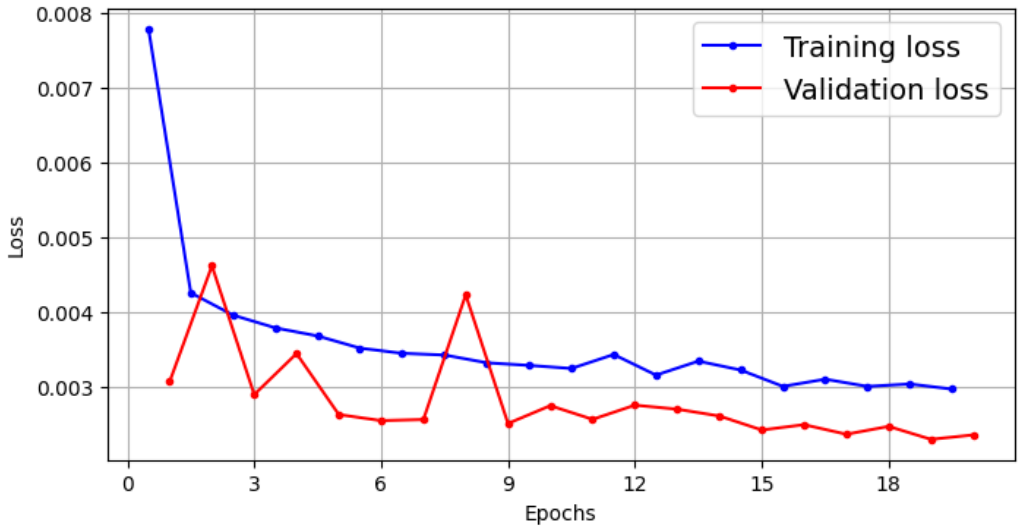

```
51/51 [==============================]
m1_GRU:  0.0037777069956064224
```

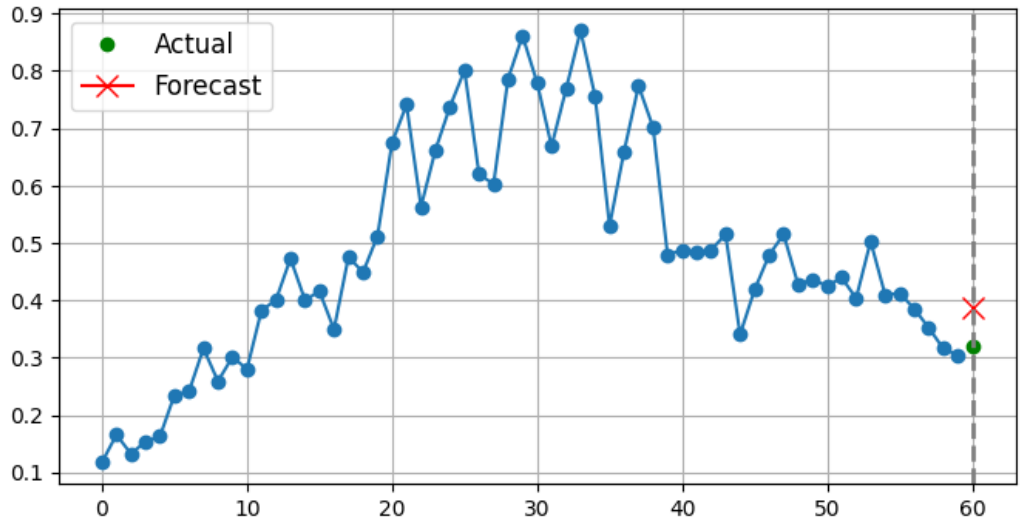




6) many-to-one with CNN1D

```python
def model_conv1d():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid', activation='relu',
               input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        Flatten(),
        Dense(32, activation='relu'),
        Dense(look_forward,  activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```



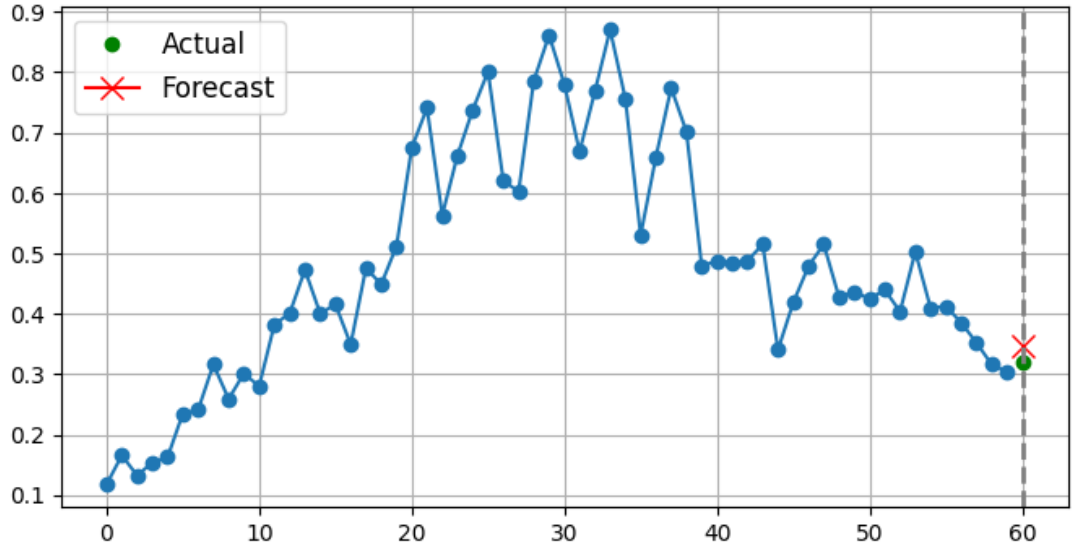


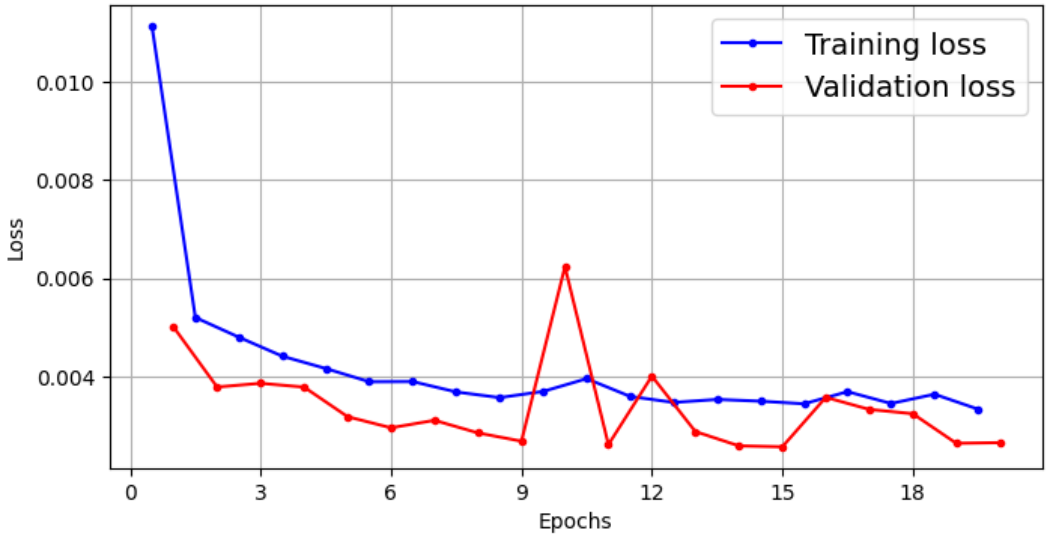

```python
a6, _ = model.evaluate(X_test, y_test)
print('m1_Conv1D: ', a6)
```

```
51/51 [==============================] -
m1_Conv1D:  0.0032061387319117785
```

7) many-to-one with LSTM-CNN1D

```
def model_cnnlstm():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid',
               activation='relu', input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        LSTM(32, return_sequences=True),
        Flatten(),
        Dense(32),
        Dense(look_forward, activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```






```
a7, _ = model.evaluate(X_test, y_test)
print('m1_Hybrid: ', a7)
```

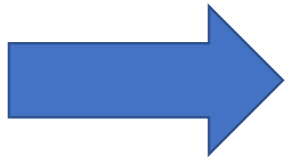
```
51/51 [==============================]
m1_Hybrid:  0.0033321112859994173
```

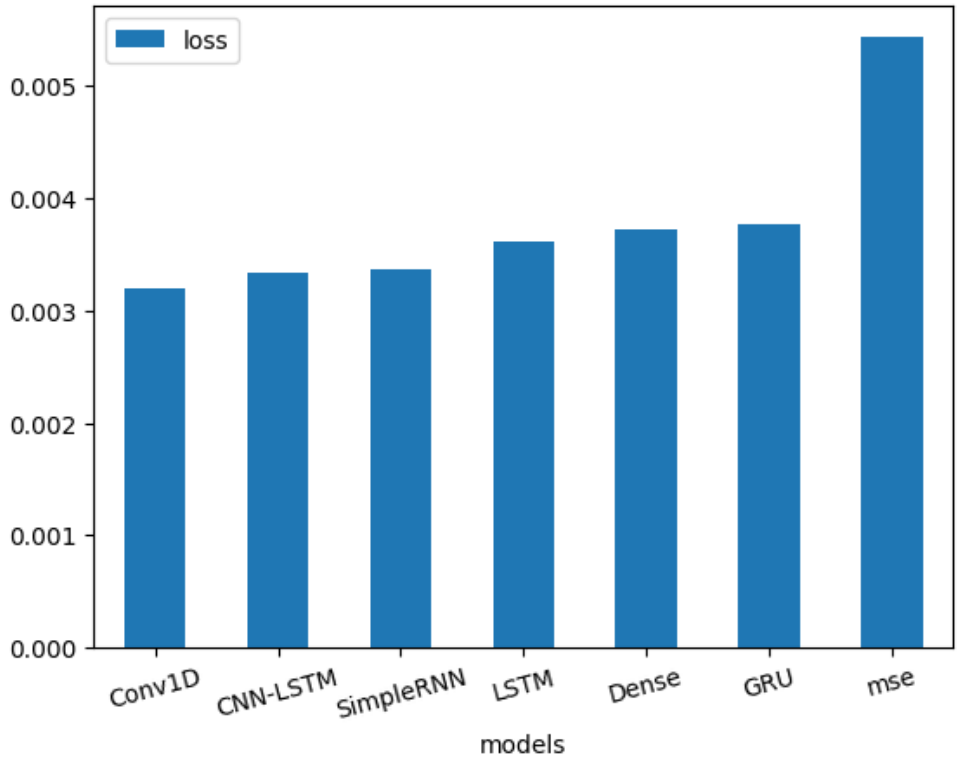
Summary I : Many to One Applications

```
df = pd.DataFrame({
    'models': ['mse', 'Dense', 'SimpleRNN','LSTM','GRU', 'Conv1D','CNN-LSTM' ],
    'loss':[a1, a2, a3, a4, a5, a6, a7]})
df=df.sort_values(by='loss', ascending=True)
```

```
df
```

	models	loss
5	Conv1D	0.003206
6	CNN-LSTM	0.003332
2	SimpleRNN	0.003366
3	LSTM	0.003614
1	Dense	0.003717
4	GRU	0.003778
0	mse	0.005438

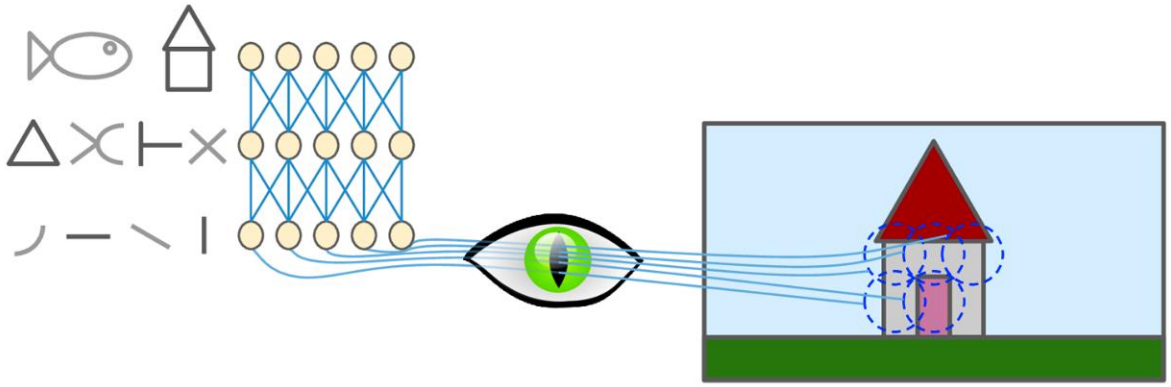




10) 2차원 CNN 모델 소개

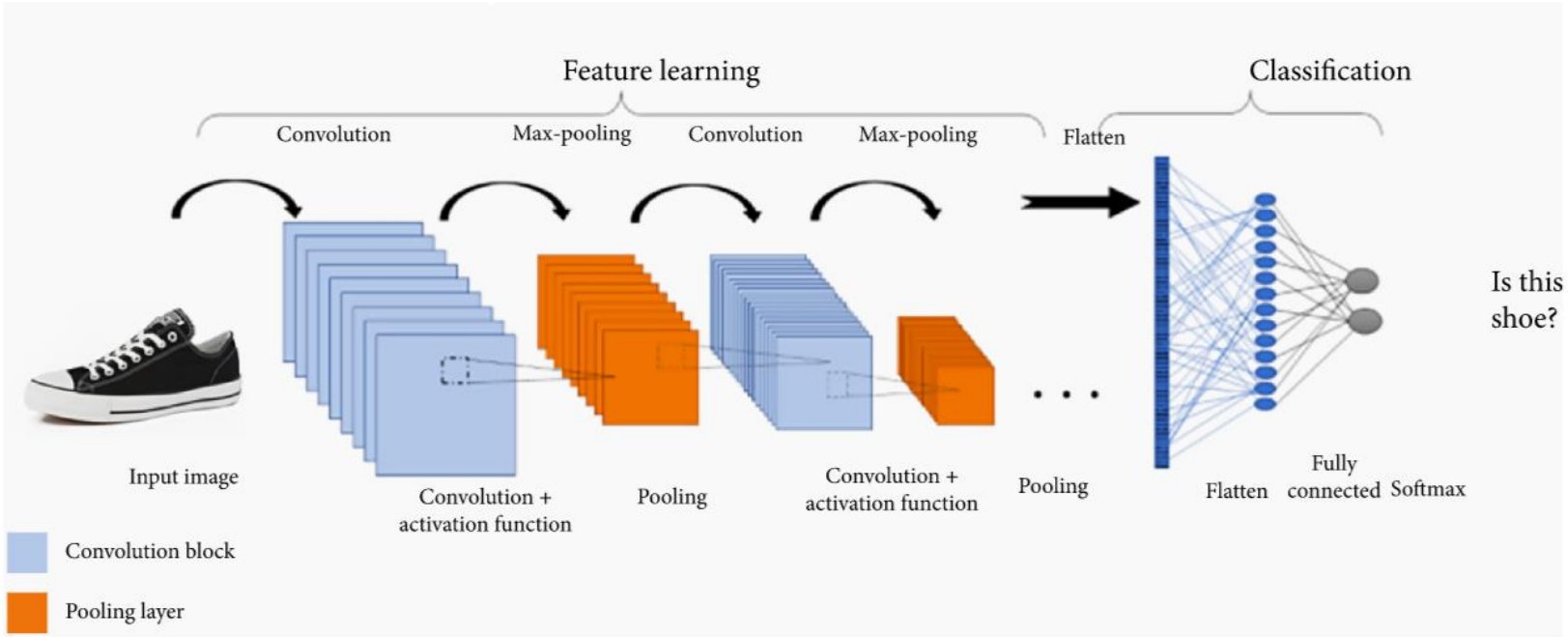
고양이의 시각피질

- Hubel과 Wiesel은 시각피질의 세포들을 여러 유형으로 구별 (1959)



합성곱 신경망은 뉴런 사이의 연결 패턴이 동물 시각 피질의 조직과 유사하다는 점에 영감을 받았다. 개별 피질 뉴런은 수용장(receptive field)으로 알려진 시야의 제한된 영역에서만 자극에 반응한다. 상이한 뉴런의 수용 필드는 전체 시야를 볼 수 있도록 부분적으로 중첩된다.[출처 필요]

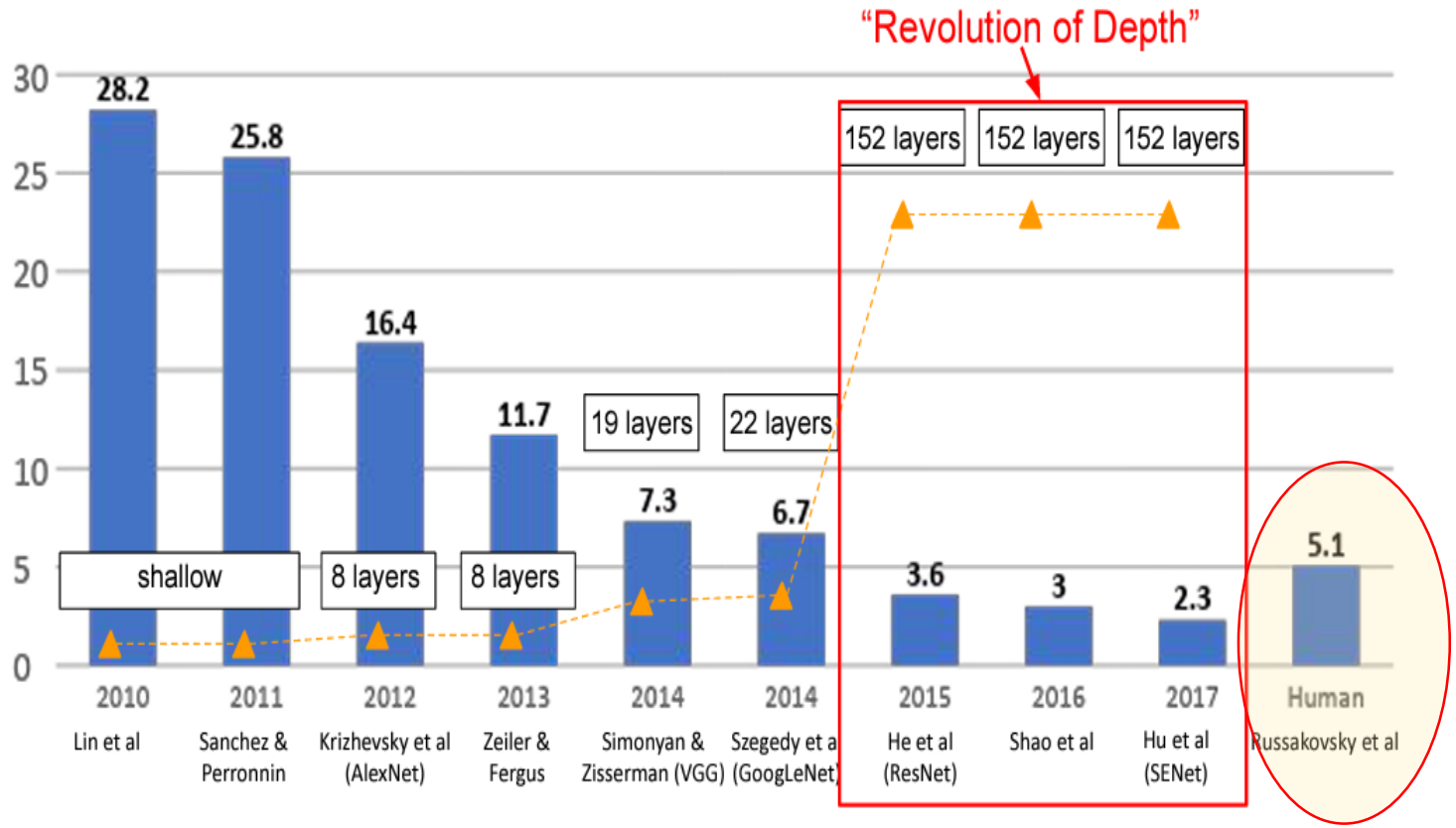
CNNs for Image Classification



source : https://www.hindawi.com/journals/wcmc/2022/7549397/fig2/

Main Breakthrough for CNNs: AlexNet & ImageNet

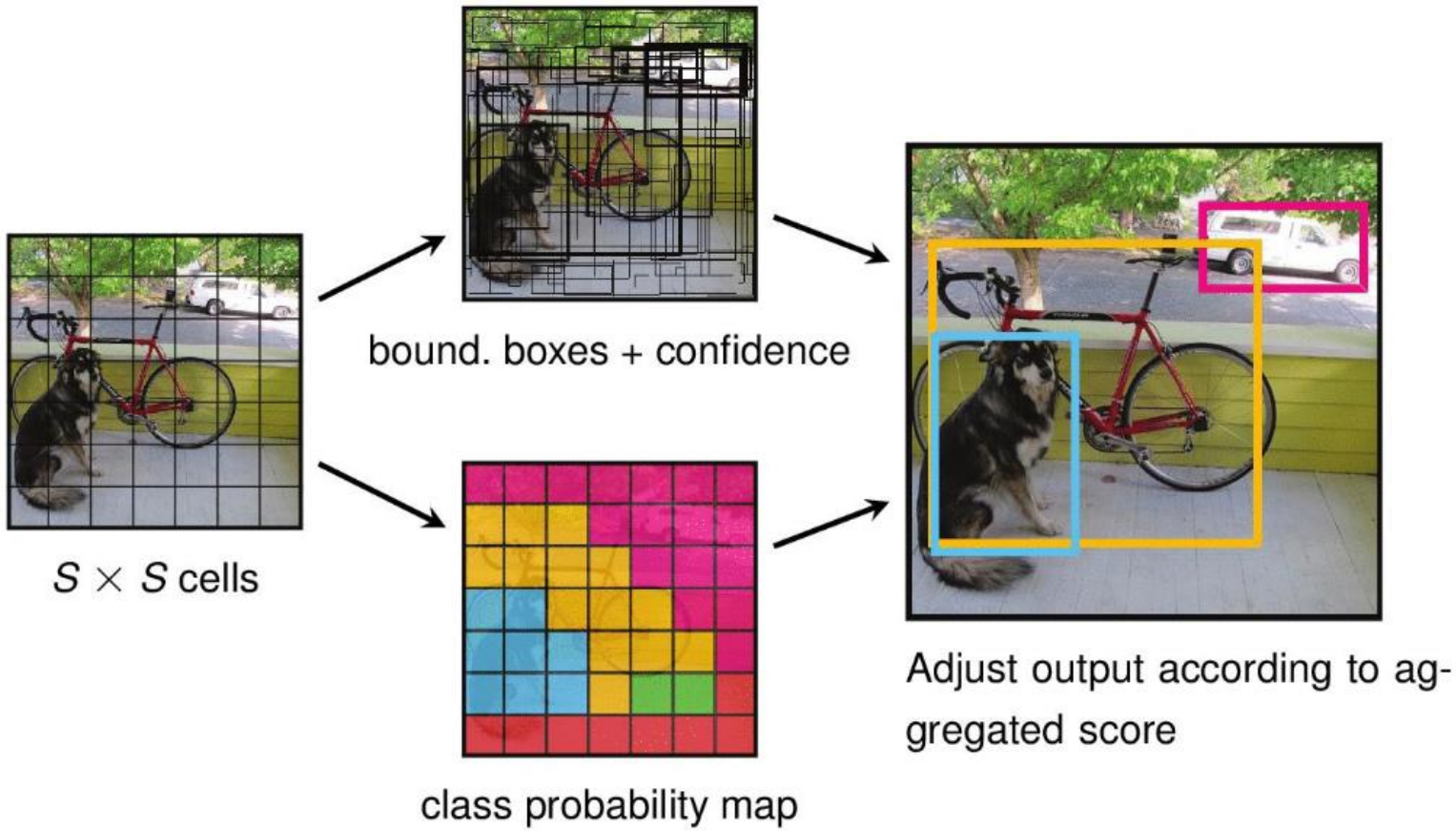
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
  - ✓ Human Error ~ 5.1%



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

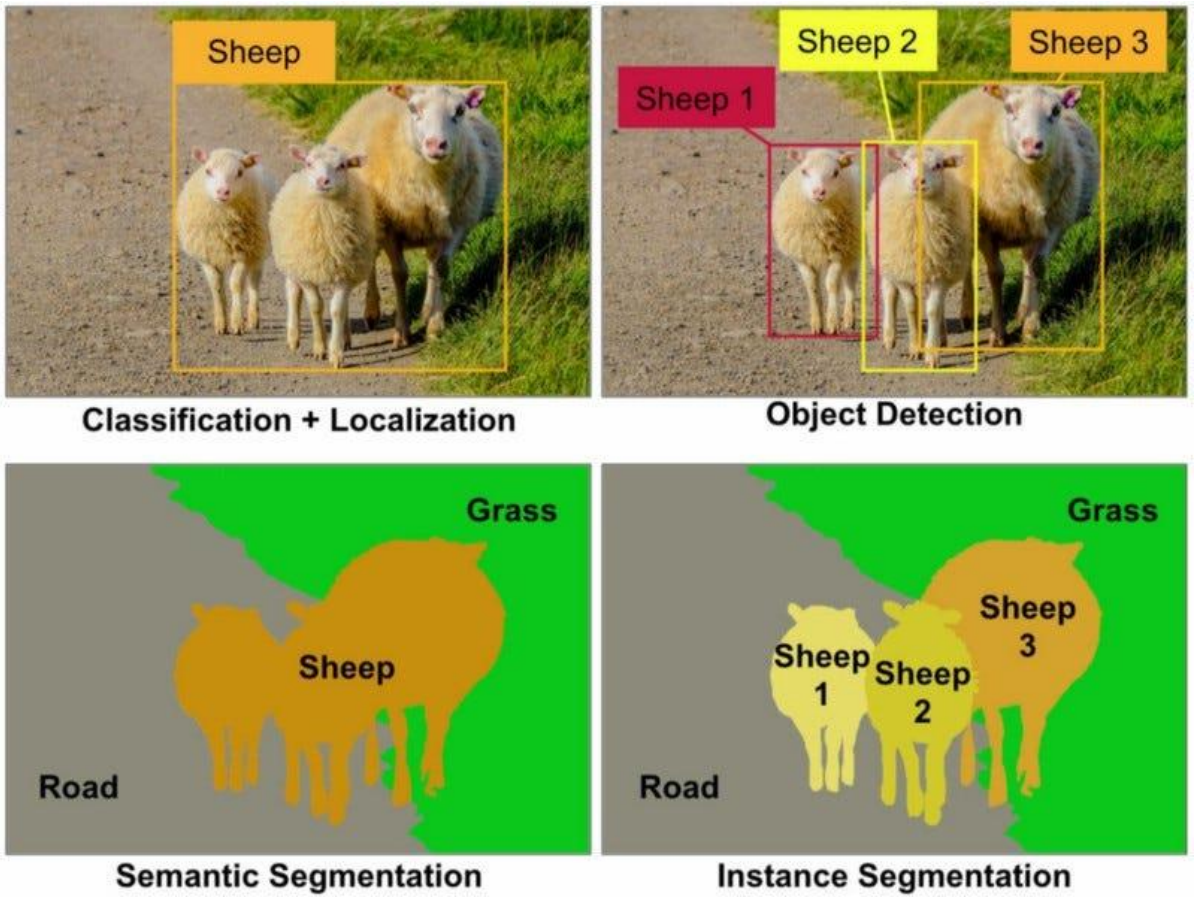
CNN applications: Object Detection and YOLO

You Only Look Once (YOLO) Algorithm



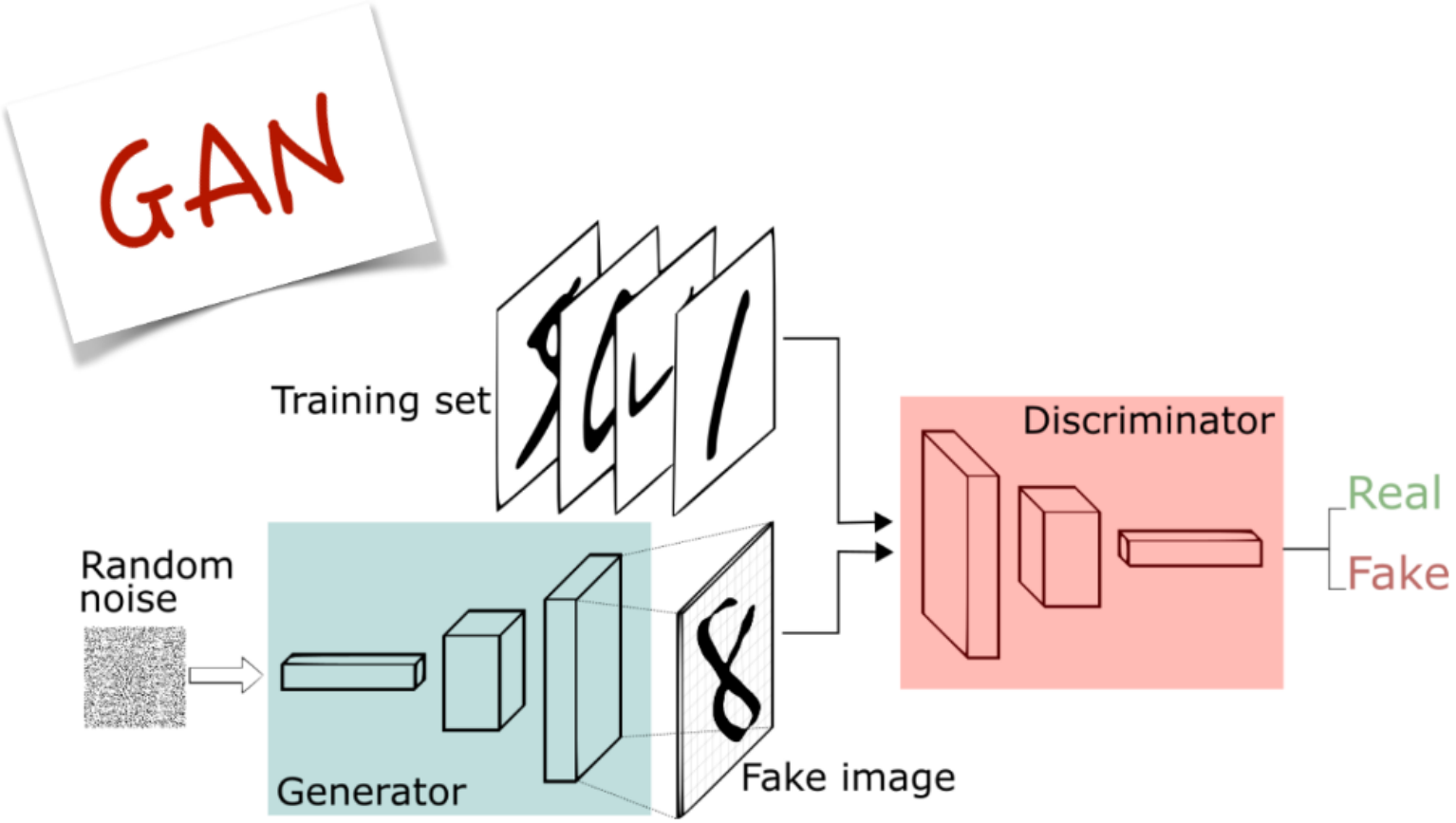
source : https://lme.tf.fau.de/lecture-notes/lecture-notes-dl/lecture-notes-in-deep-learning-segmentation-and-object-detection-part-4/

CNN applications: Object Segmentation



source https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50

CNN applications: Generative Adversarial Networks



source : https://velog.io/@hwany/GAN

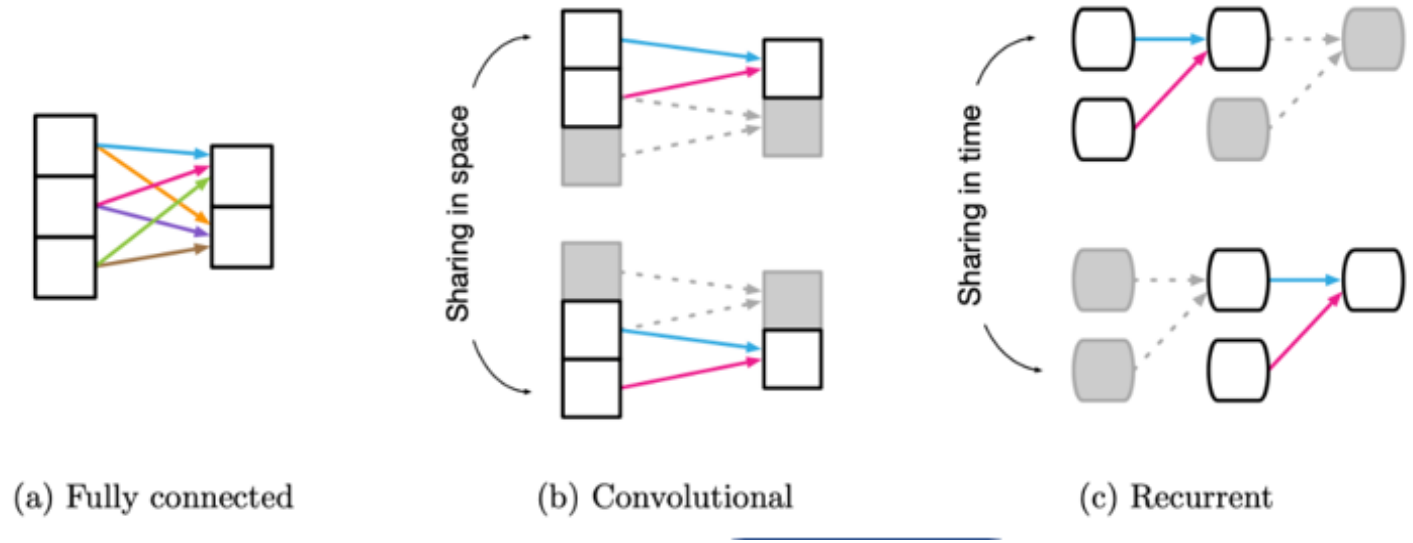
❖ 딥런닝에서 관계 귀납 편향(Relational inductive biases)

학습시 정확한 예측을 위해 추가적인 가정을 하는 것

Weak Relation

Locality

Sequential



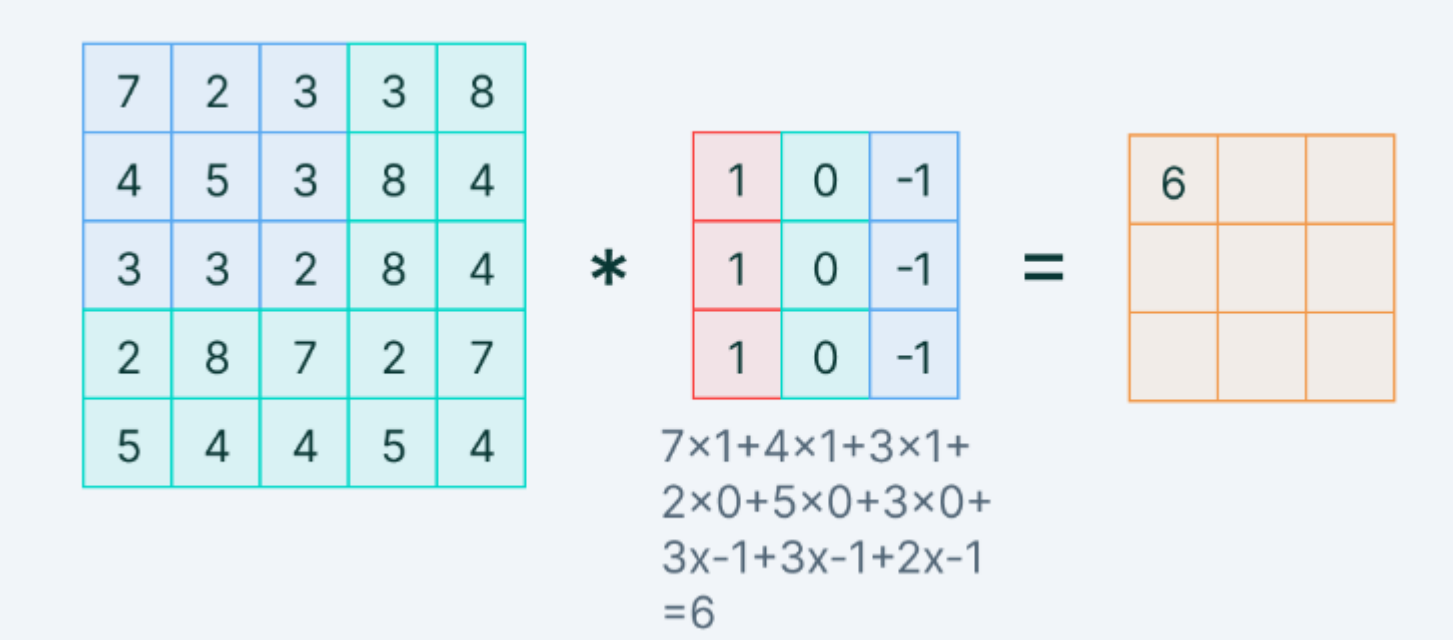
(a) Fully connected (b) Convolutional (c) Recurrent

source : https://www.baeldung.com/cs/ml-inductive-bias

CNN works by comparing images piece by piece

❖ Convolution Operation (합성곱 연산)

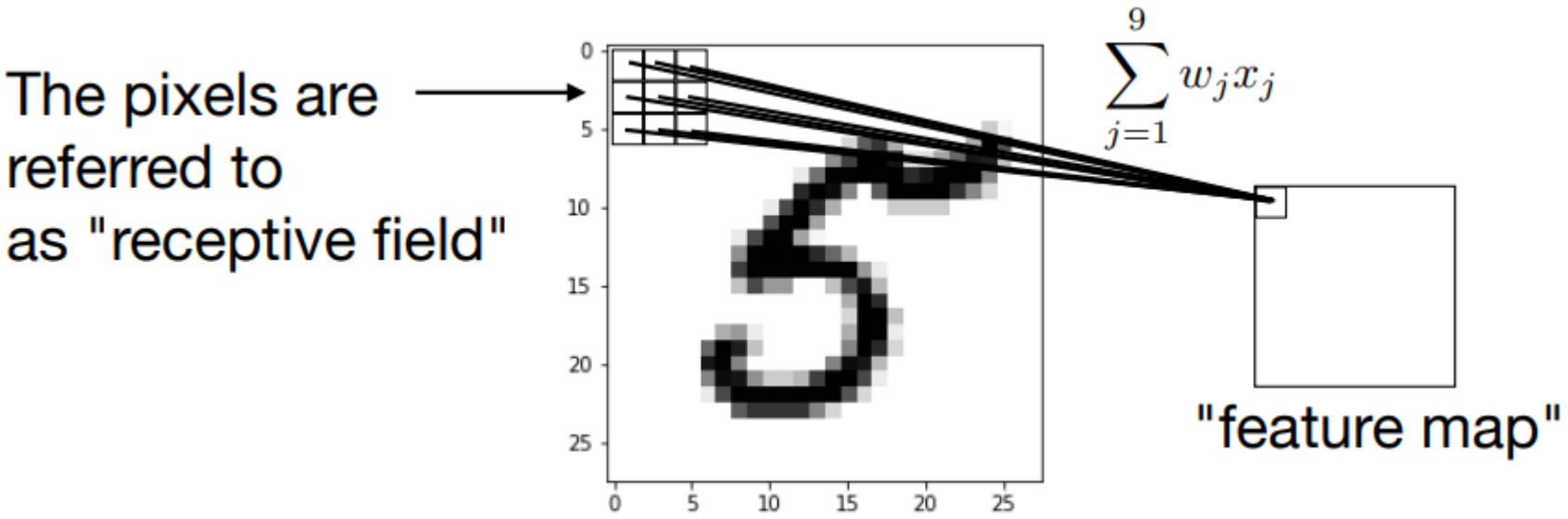
- ✓ Element-wise multiplication between the filter-sized patch of the input image
- ✓ And filter is done, which is then summed.



(source) https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html

RNN 처럼 Weight Sharing

- A filter(kernel) slides over the inputs to generate a feature map

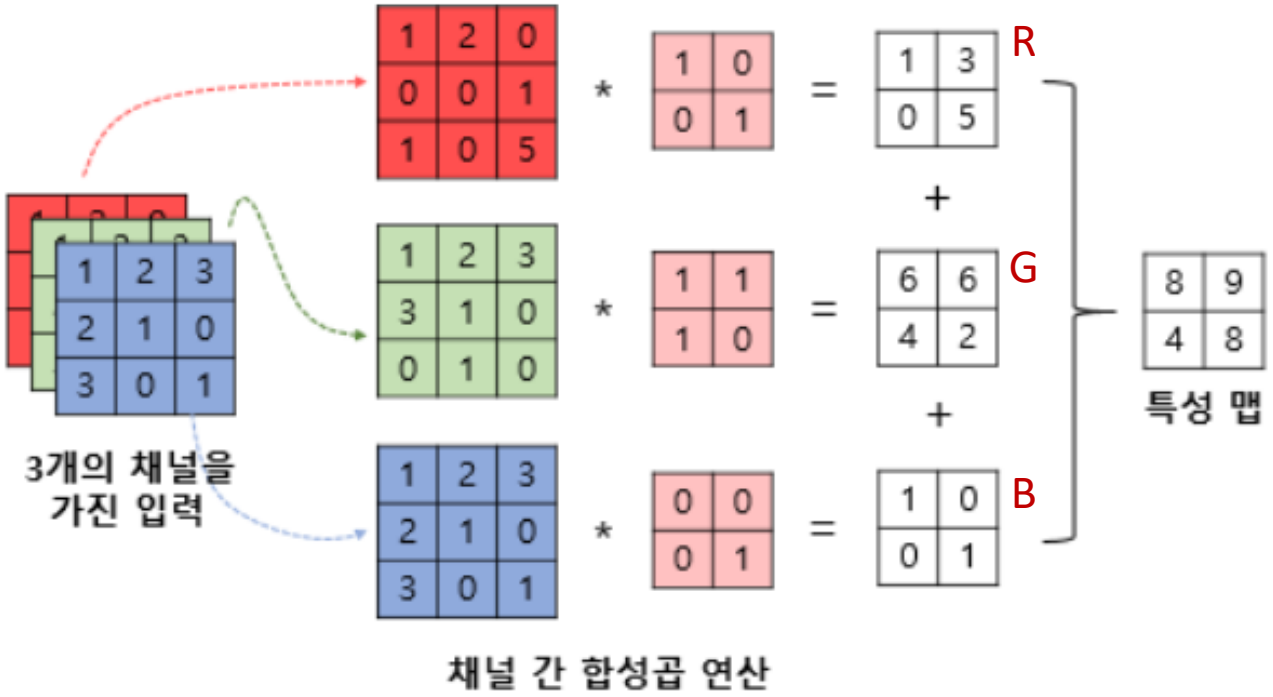


source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

Convolutions with Color Channels

- For multiple channels, the feature map does not contain RGB meanings

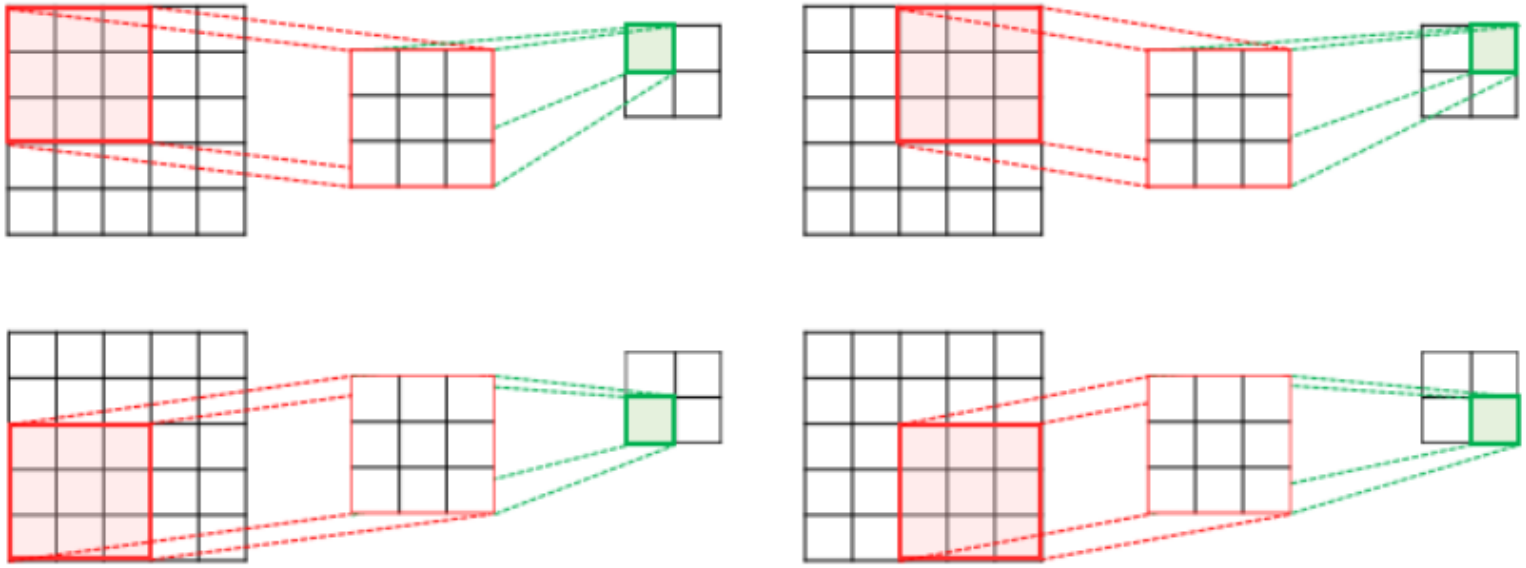
Channel is Color
Channel : RGB
Value: 0~255



(source) https://wikidocs.net/64066

Strided Convolution

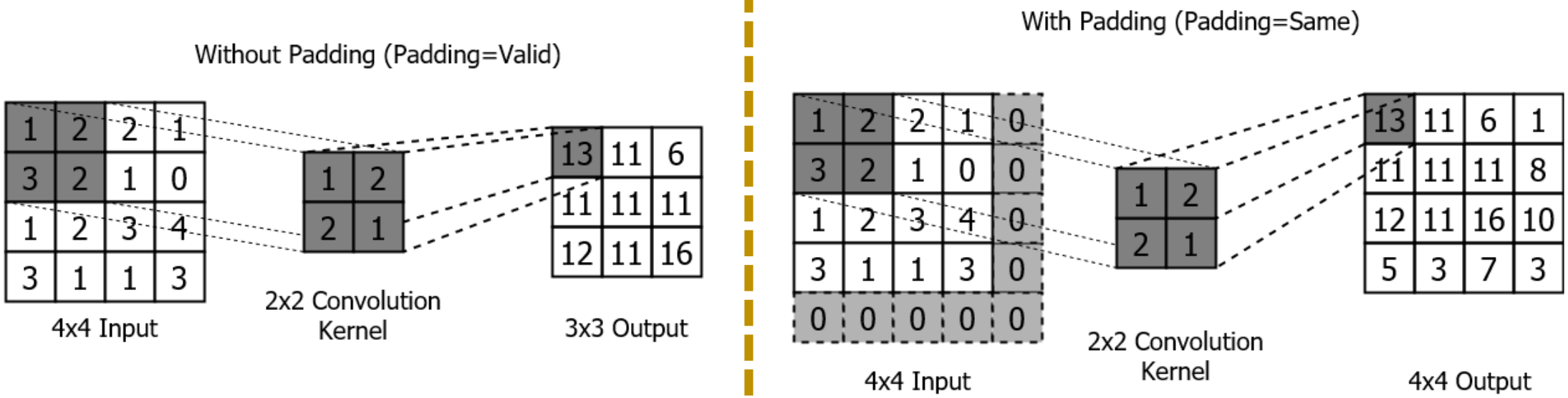
- Stride is range of movement
  - ✓ 5 × 5 input (spatially) assume 3x3 filter applied with stride,
  - ✓ and finally 2x2 size feature map



(source) https://wikidocs.net/64066

Padding basically extends the area of an image

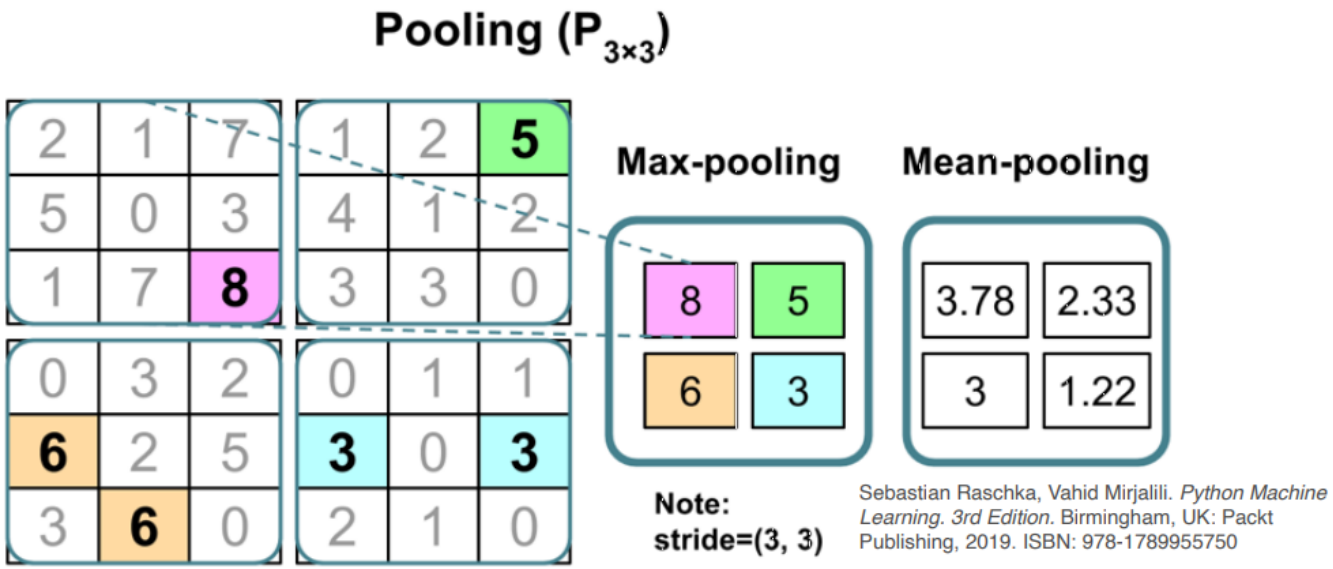
- Zero padding fills zero
  - ✓ padding="valid", padding="same"



(source) https://livebook.manning.com/book/tensorflow-in-action/chapter-4/v-1/

Pooling Layers Can Help With Local Invariance

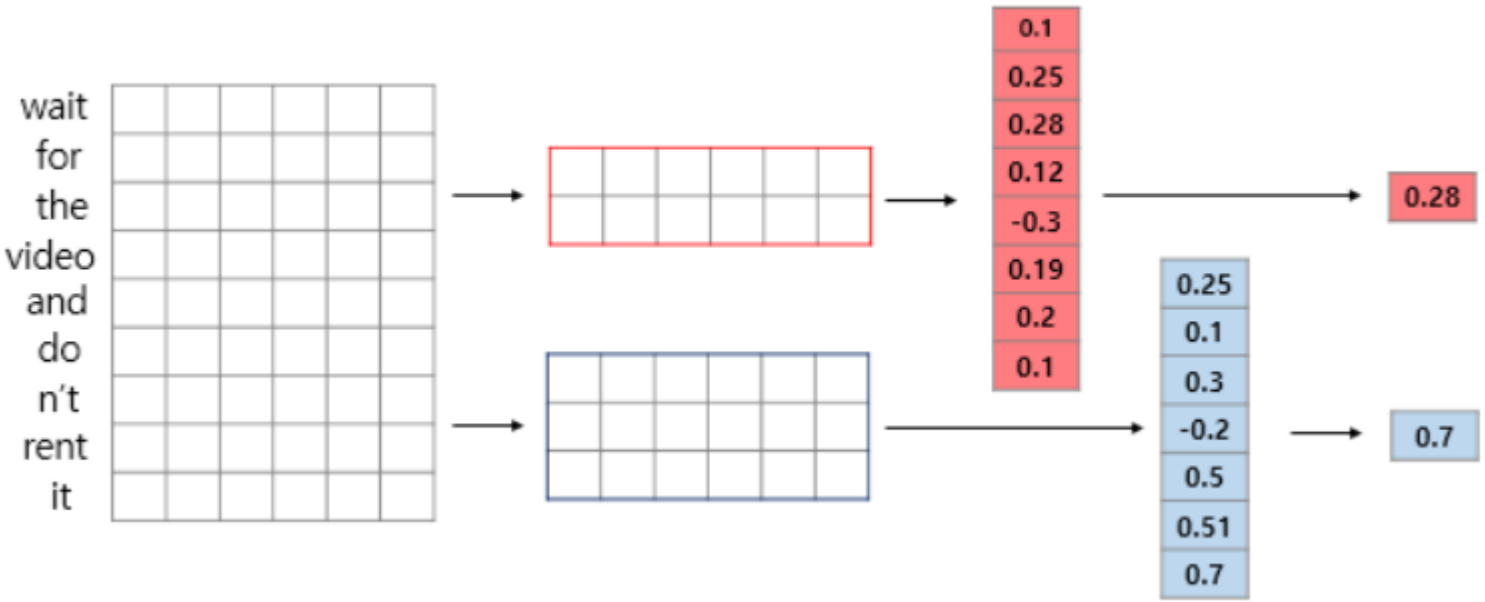
- Pooling : information is lost.
  - ✓ relative position is important (like face recognition)



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

1D CNN에서 Max Pooling

- 1D CNN 예시
  - ✓ 합성곱+활성함수 다음에 풀링층을 적용
  - ✓ 맥스풀링 사용 (커널크기는 2 혹은 3)
- 커널 너비는 특성(feature)의 길이이고 커널의 높이는 1차원 CNN의 커널 크기



입력 데이터 다운로드

```
import tensorflow as tf
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('https://raw.githubusercontent.com/hongsukyi/Lectures/main/vds/vds.csv')
df.head(2)
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84

속도 예측을 위한 정규화

```
df_uni = df.iloc[:,5:6].values
```

```
df_uni
```

```
array([[50.3],
       [58.9],
       [50.6],
       ...,
       [50.6],
       [59.3],
       [52.5]])
```

```
print(df_uni.shape)
```

```
(8064, 1)
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range = (0,1))
scaled_df = scaler.fit_transform(df_uni)

scaled_df
```

```
array([[0.52350699],
       [0.63278272],
       [0.52731893],
       ...,
       [0.52731893],
       [0.63786531],
       [0.55146125]])
```

```
look_back = 12*24
```

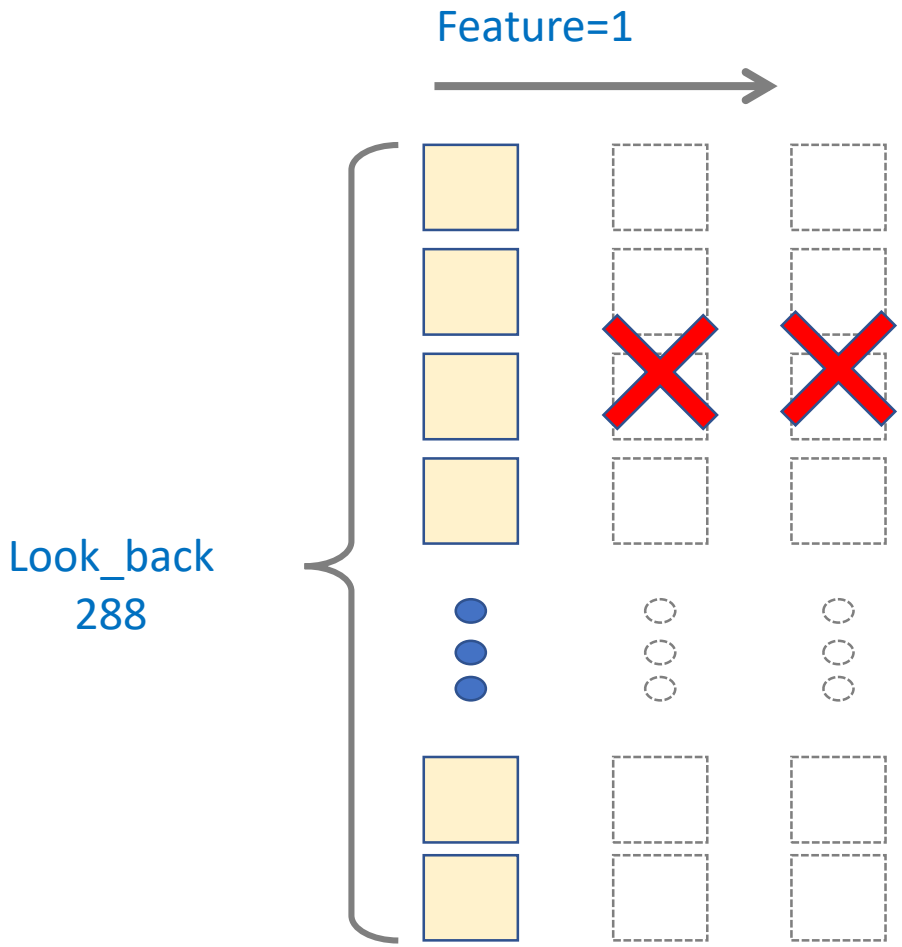
RNN 처럼 Conv1d를 위한 데이터 구조:

```
look_back = 12*24
```

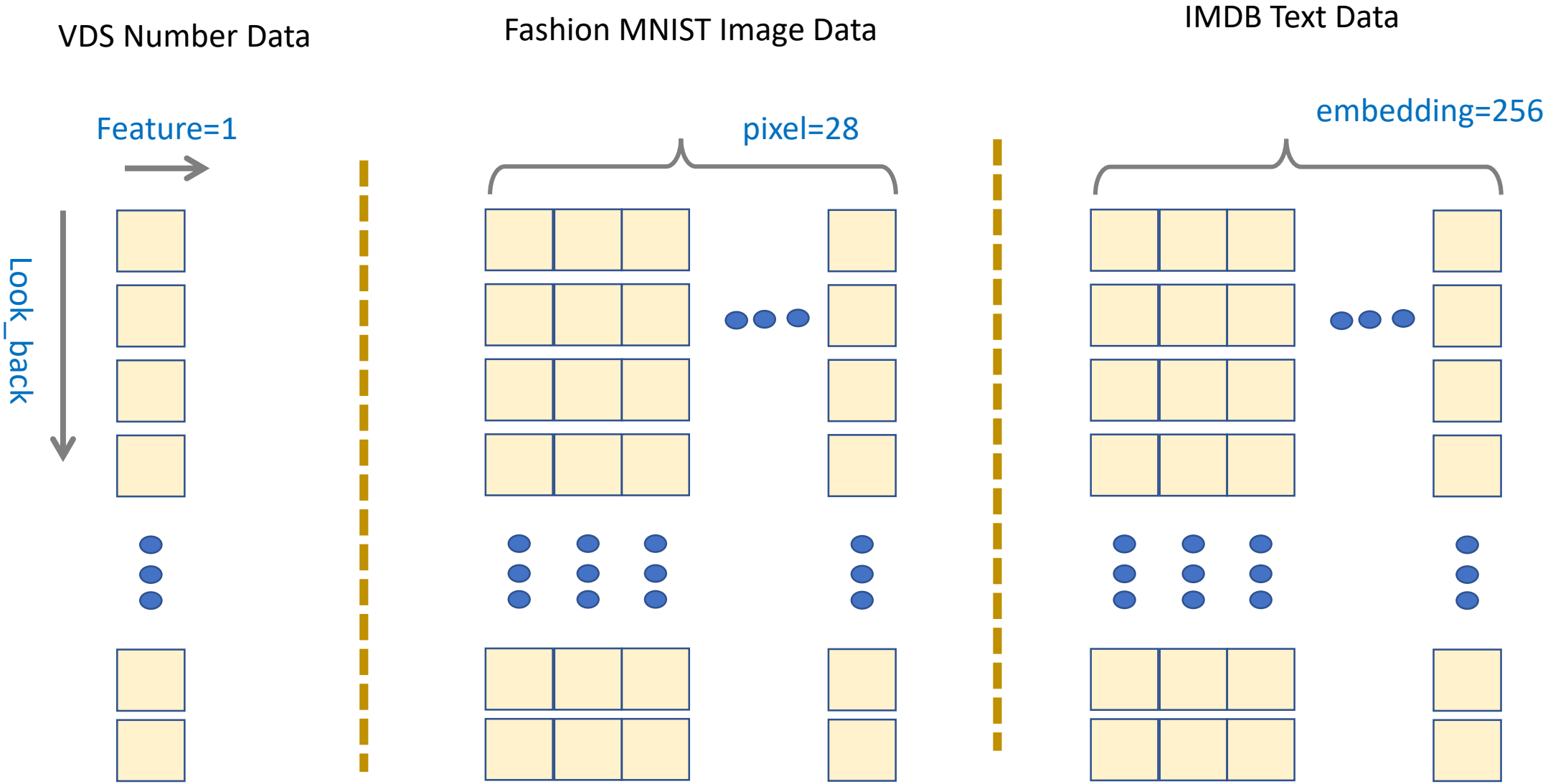
```
X = []
y = []
for i in range(len(scaled_df)-look_back-1):
    X.append(scaled_df[i:(i+look_back)])
    y.append(scaled_df[(i+look_back)])

X = np.array(X)
y = np.array(y)
```

```
print(X.shape, y.shape)
```

```
(7775, 288, 1) (7775, 1)
```



숫자, 그림, 텍스트 데이터 : batch_size=1일 경우



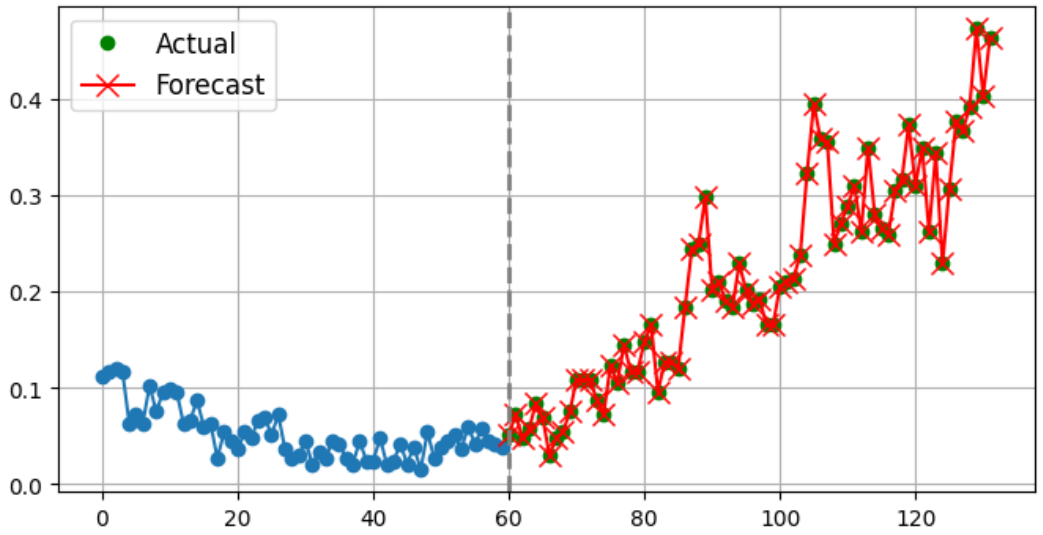
교통흐름 예측 모델 (II)
Multi-step ahead prediction

Multi-step ahead prediction model

```
look_back = 12*5
look_forward = 12*6
```

```
X, y = create_dataset(df_scaled,look_back,look_forward)
```

```
plot_windows(X,y,y)
```




```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size=0.20, shuffle=False)

print('X_train:', X_train.shape)
print('X_test :', X_test.shape)
print('y_train:', y_train.shape)
print('y_test :', y_test.shape)
```

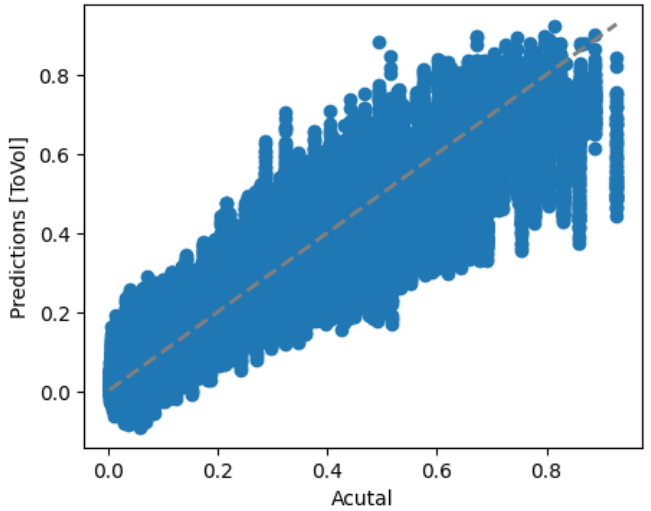
```
X_train: (6346, 60, 1)
X_test : (1587, 60, 1)
y_train: (6346, 72, 1)
y_test : (1587, 72, 1)
```

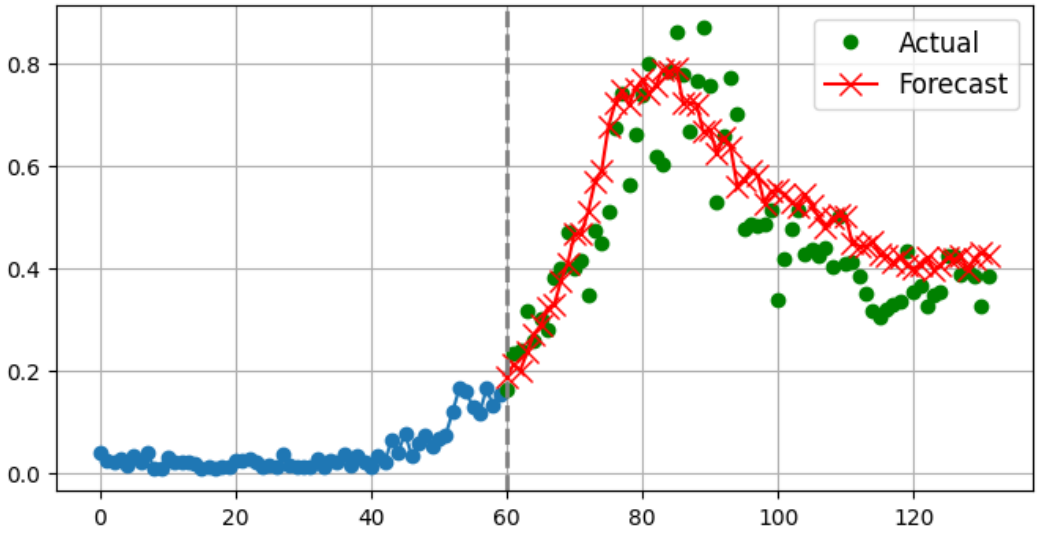
```
def model_dnn():
    model = Sequential([ Flatten(input_shape=[look_back,1]),
                         Dense(100, activation='relu'),
                         Dense(look_forward) ])
    optimizers = tf.keras.optimizers.Adam(learning_rate=0.01)
    model.compile(loss="mse", optimizer=optimizers,metrics =['mae'])
    model.summary()
    return model
```

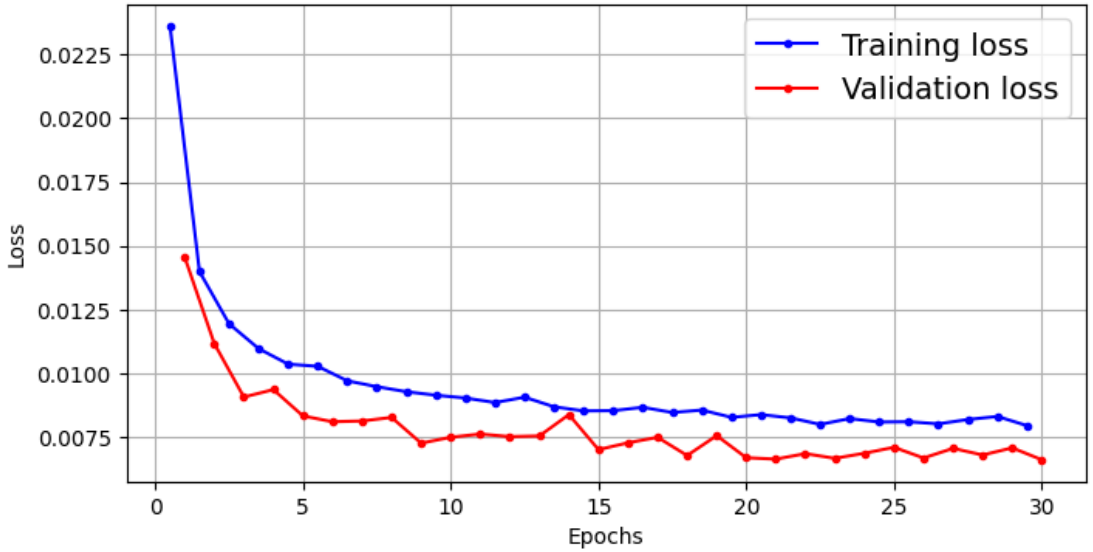
(1) m2: many-to-many with DNN

```
model = model_dnn()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 60)                0

 dense_7 (Dense)             (None, 32)                1952

 dense_8 (Dense)             (None, 32)                1056

 dense_9 (Dense)             (None, 288)               9504
```

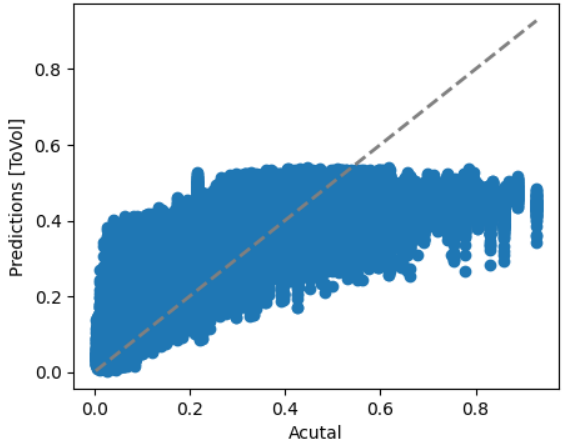


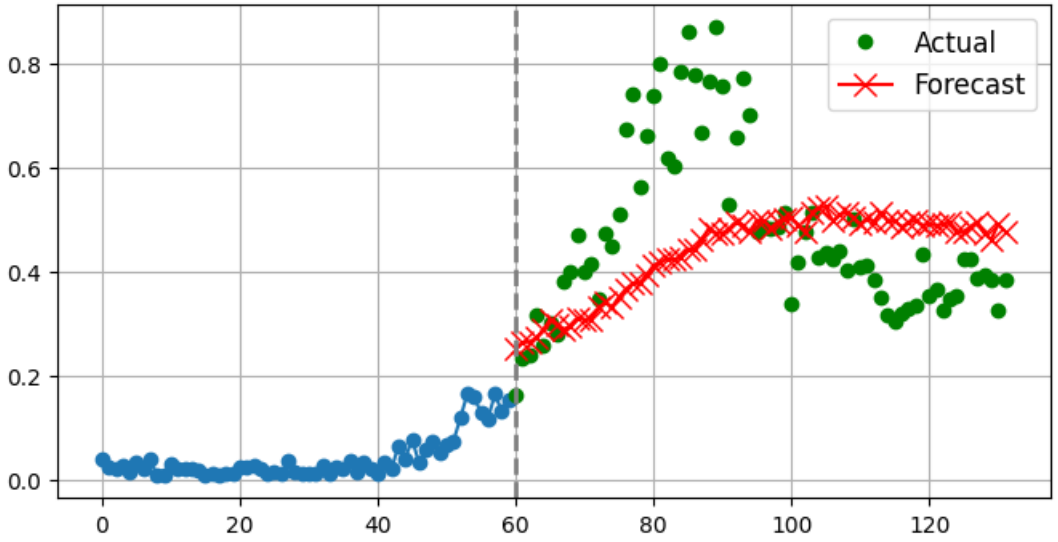


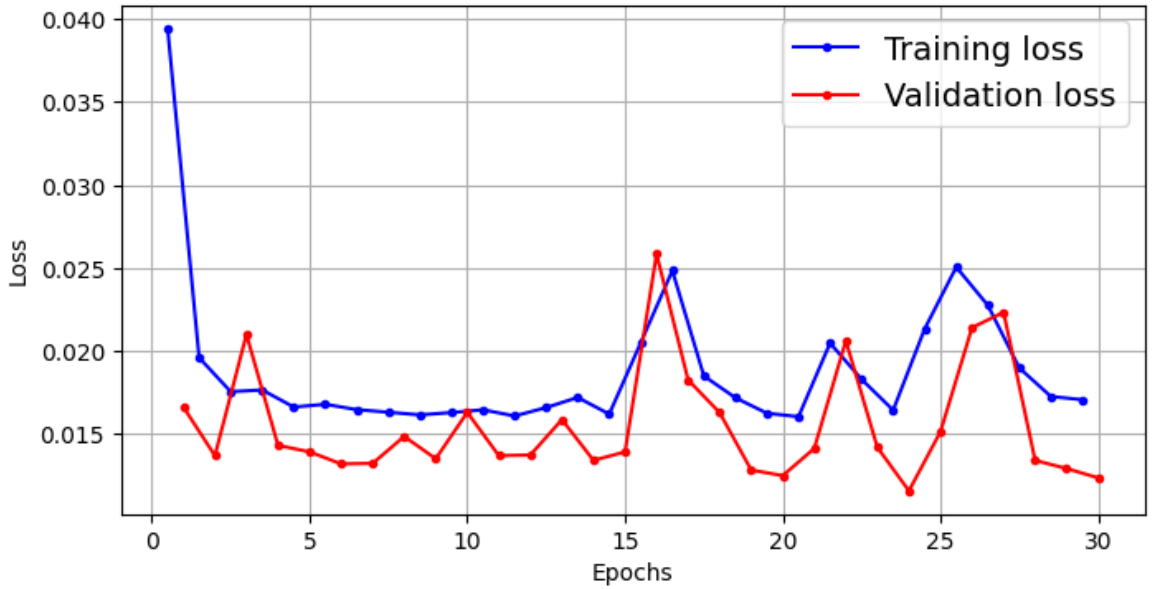


b2) : Deep SimpleRNN

```
def model_simpleRNN():
    model = Sequential([
        SimpleRNN(32, return_sequences=True, input_shape=[look_back, 1]),
        SimpleRNN(32,return_sequences=True),
        SimpleRNN(look_forward,return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

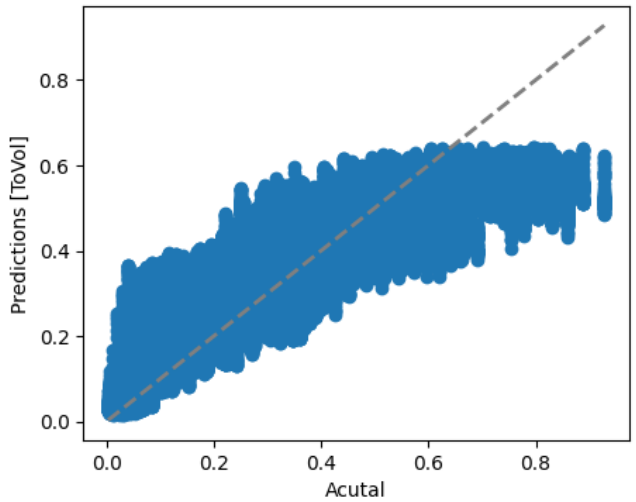


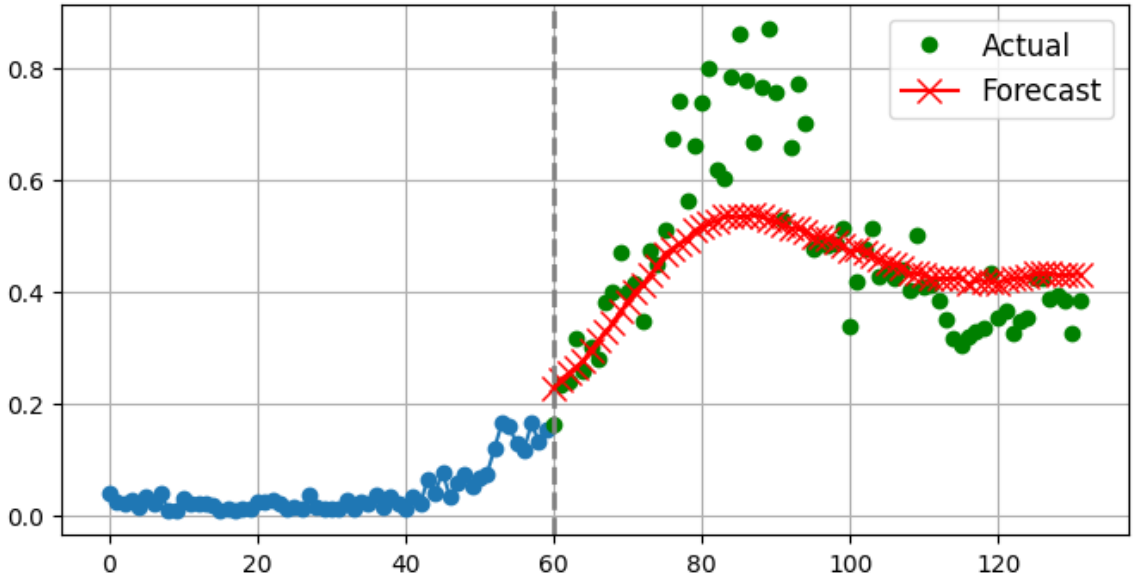


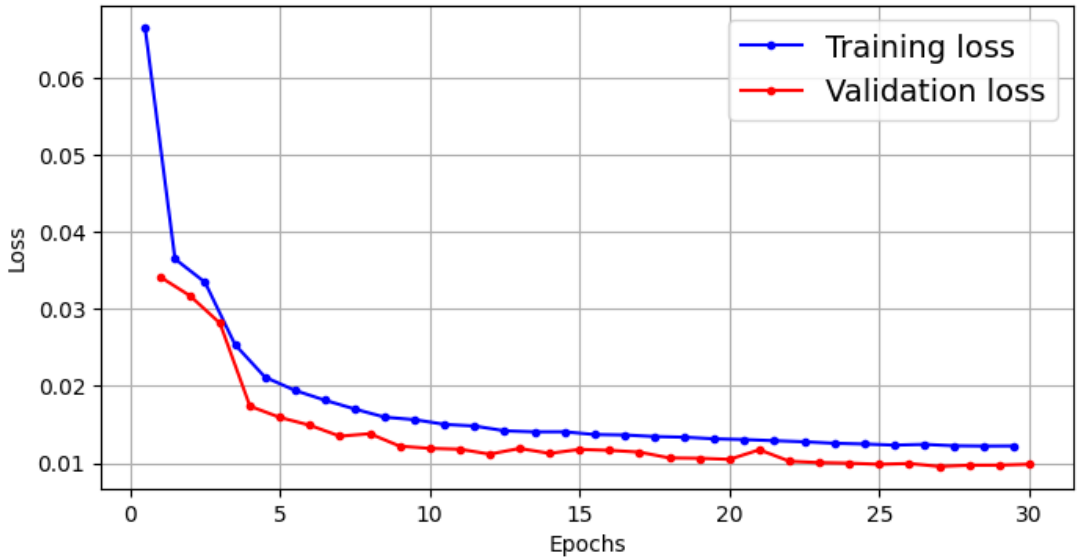


b3) many to many with LSTM

```
def model_lstm():
    model = Sequential([
        LSTM(32, return_sequences=True, input_shape=[look_back, 1]),
        LSTM(32, return_sequences=True),
        LSTM(look_forward, return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

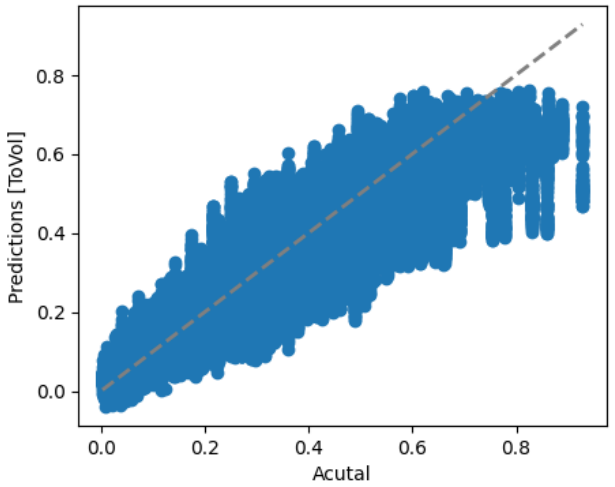


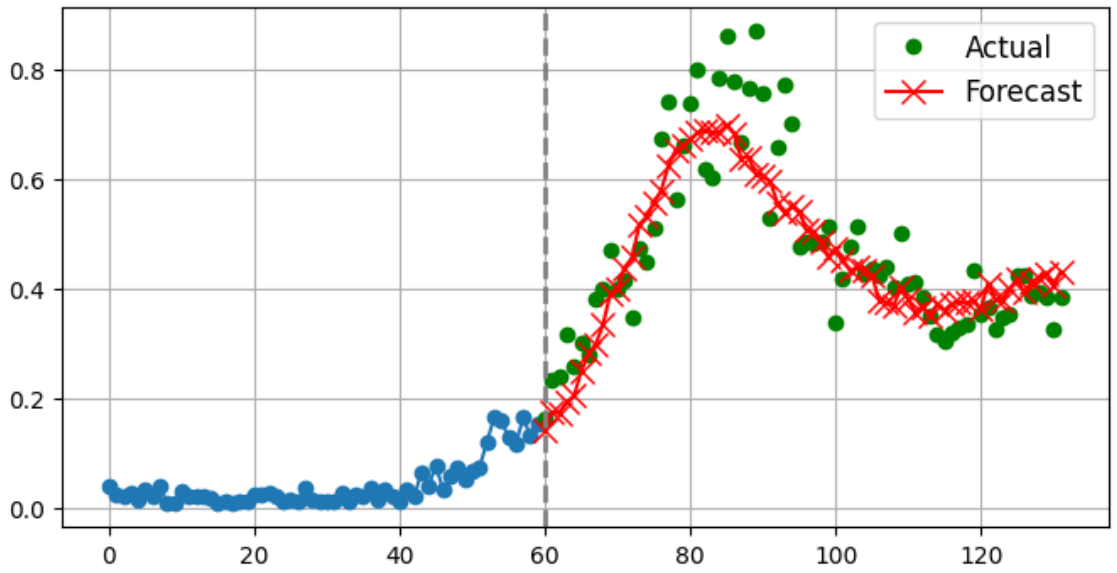


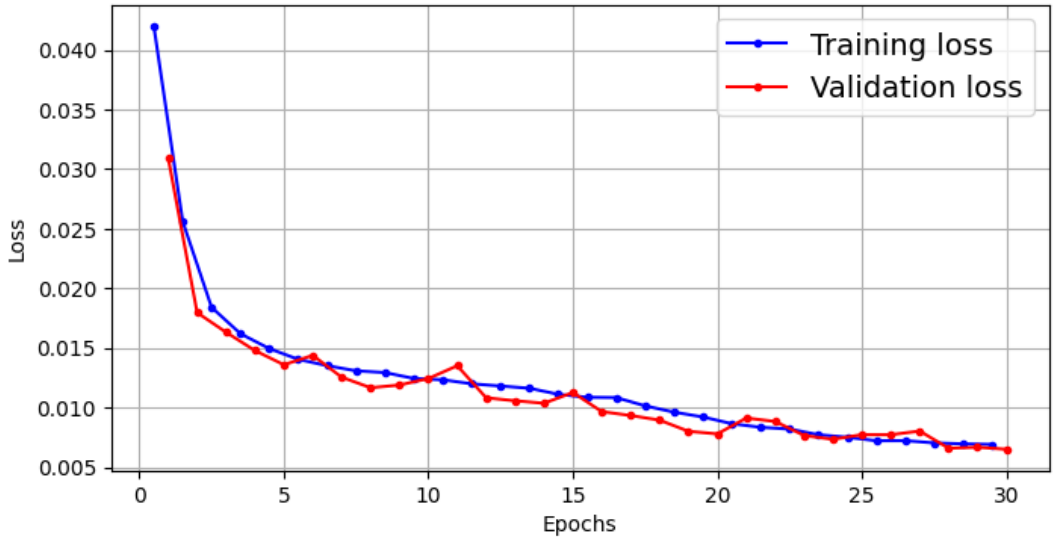


b4) many to many with GRU

```
def model_gru():
    model = Sequential([
        GRU(32, return_sequences=True, input_shape=[look_back, 1]),
        GRU(32, return_sequences=True),
        GRU(look_forward, return_sequences=False),
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

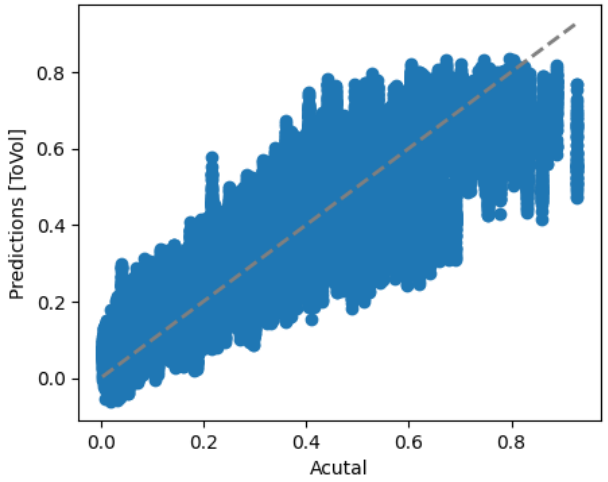


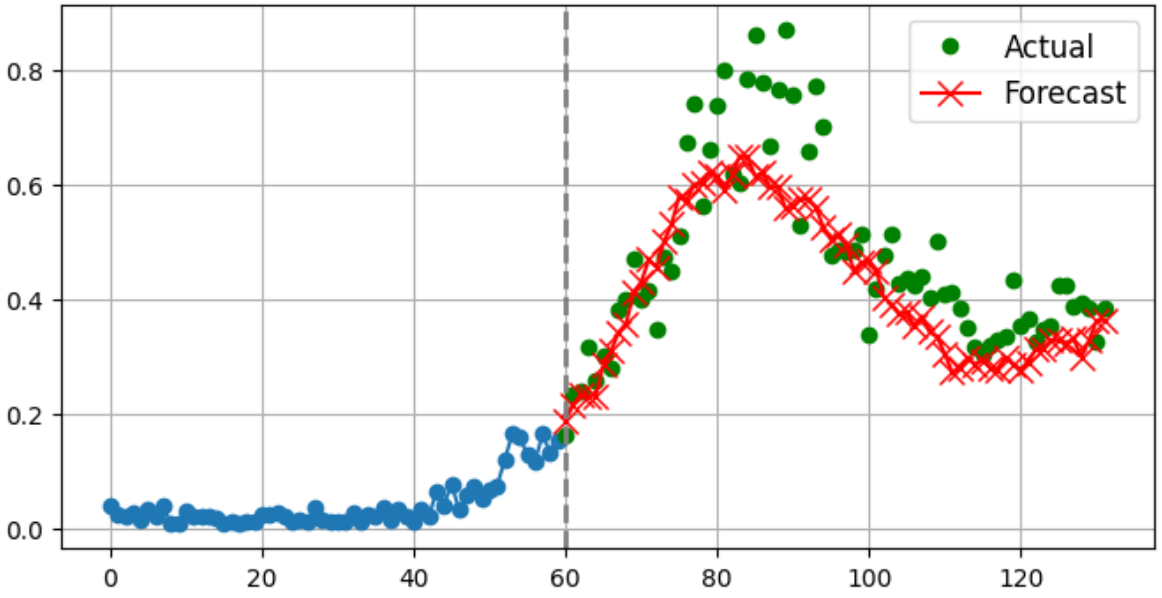


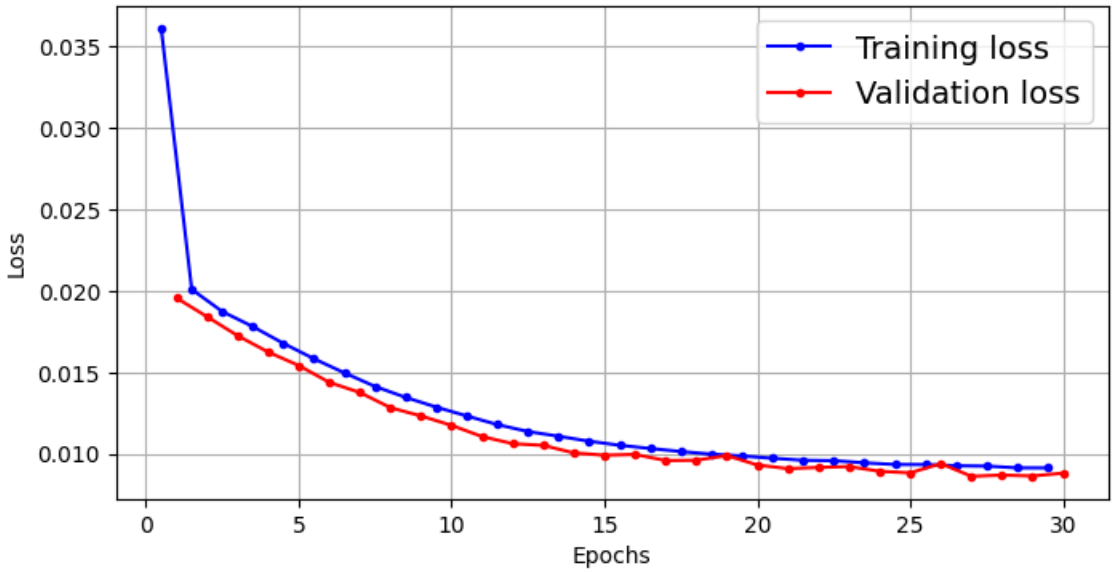


(b5) many to many with Conv1D

```python
def model_conv1d():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid', activation='relu',
               input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        Flatten(),
        Dense(32, activation='relu'),
        Dense(look_forward,  activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```

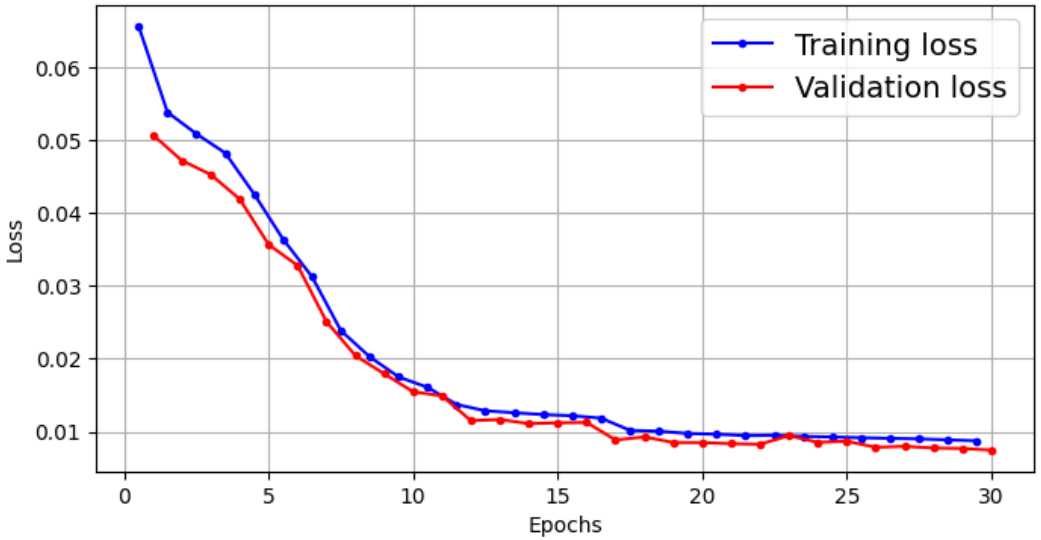


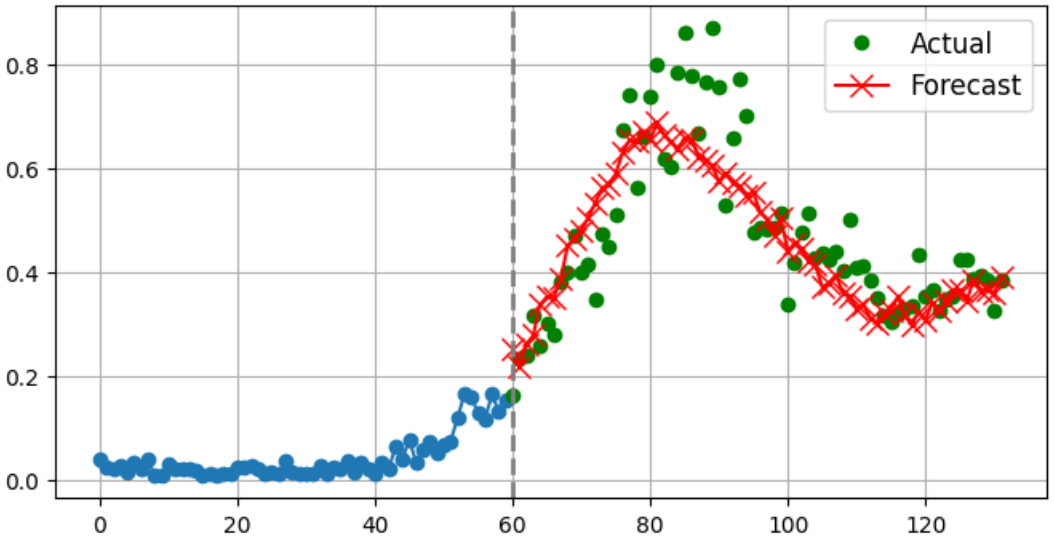


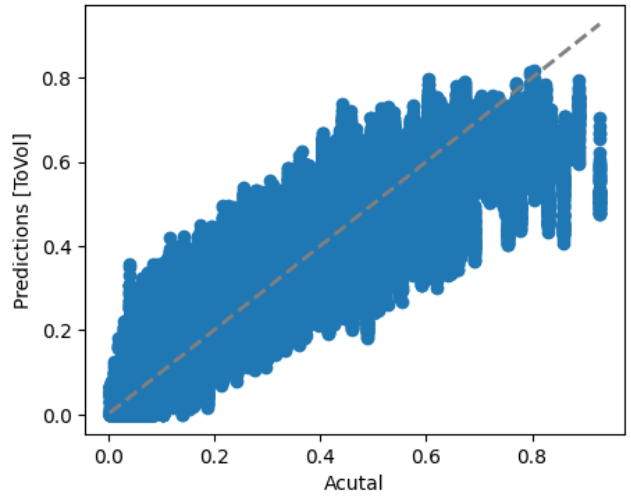


(b6) many to many with hybrid model : Conv1D+LSTM

```python
def model_cnnlstm():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid',
               activation='relu', input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        LSTM(32, return_sequences=True),
        Flatten(),
        Dense(32),
        Dense(look_forward, activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```





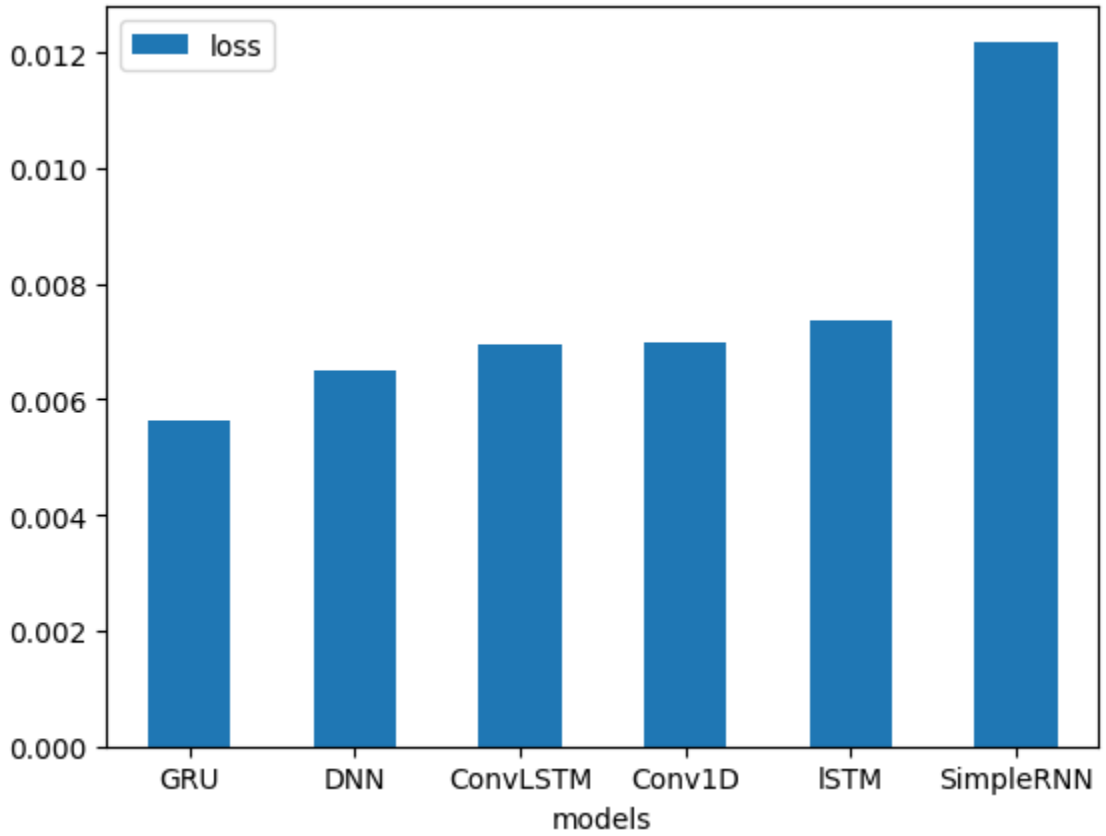


Summary of Many to Many Model

```
df = pd.DataFrame({
    'models': ['DNN', 'SimpleRNN','lSTM','GRU', 'Conv1D','ConvLSTM'],
    'loss':[b1, b2, b3, b4, b5, b6]})
df=df.sort_values(by='loss', ascending=True)
```

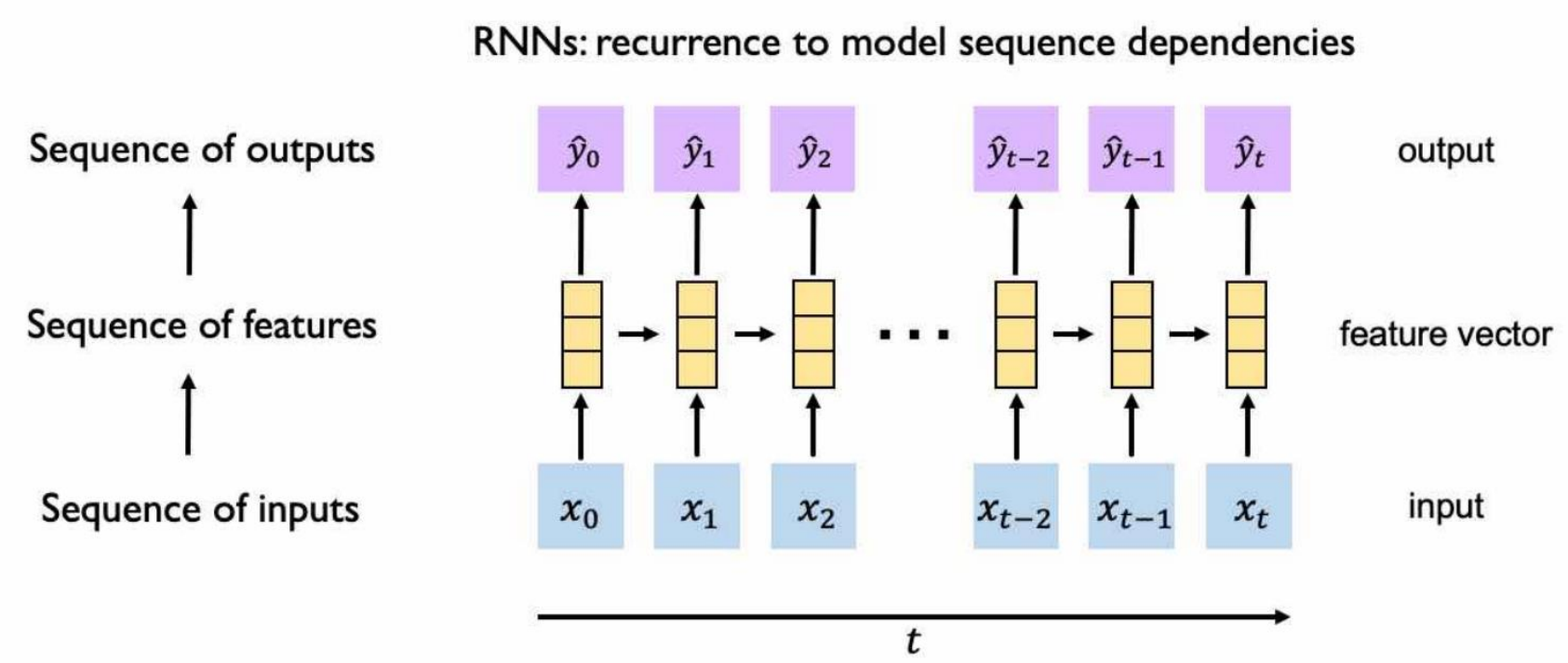
Many-to-Many model

	models	loss
3	GRU	0.005631
0	DNN	0.006497
5	ConvLSTM	0.006934
4	Conv1D	0.006991
2	lSTM	0.007380
1	SimpleRNN	0.012190



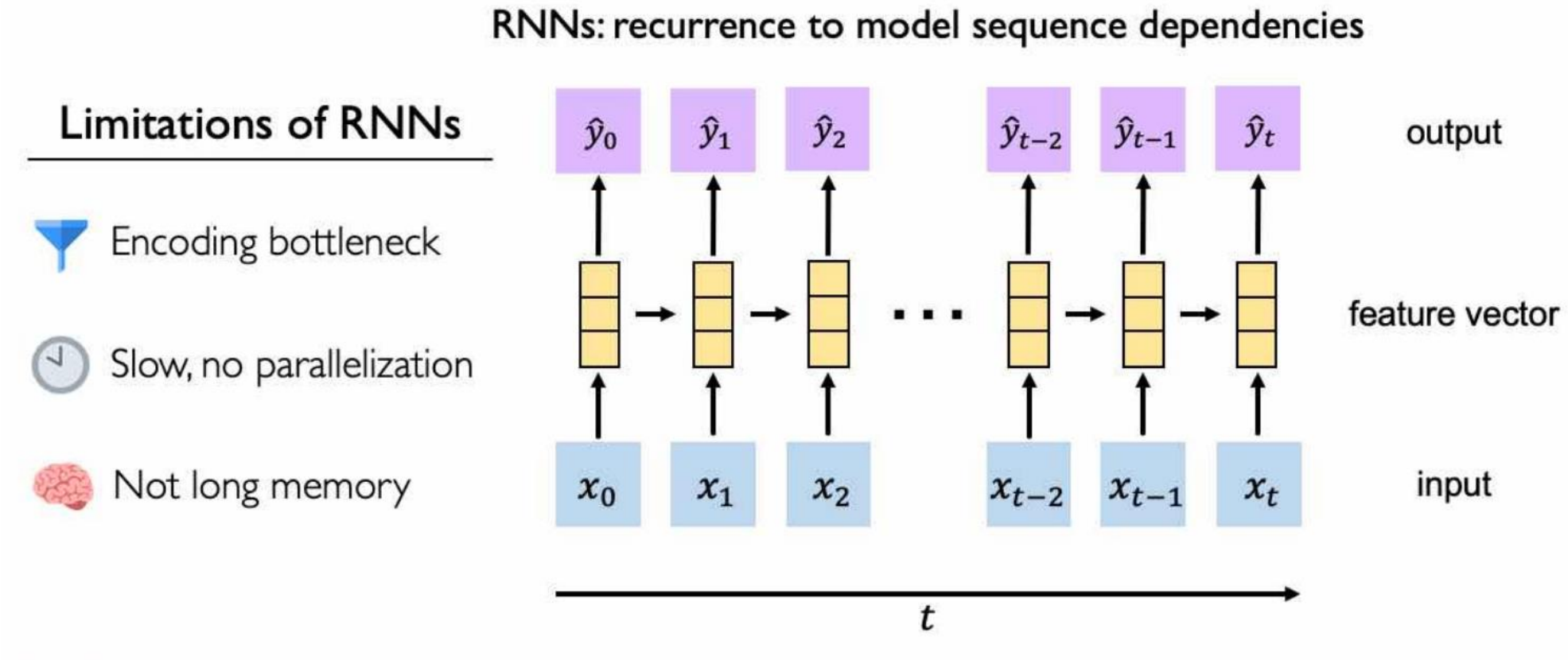
12) 멀티헤드 어텐션 및 트랜스포머 모델 소개

Goal of Sequence Modeling



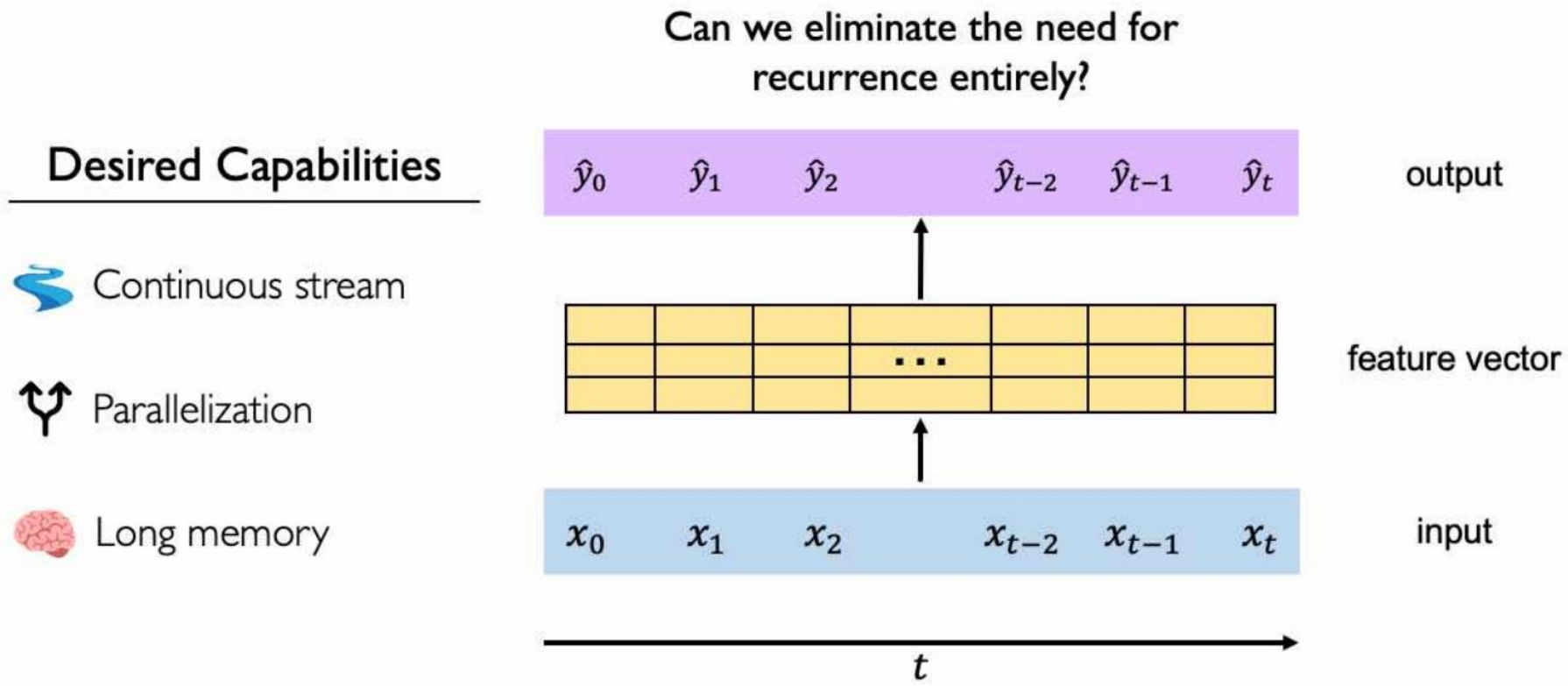
출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

Goal of Sequence Modeling



출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

Goal of Sequence Modeling



출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

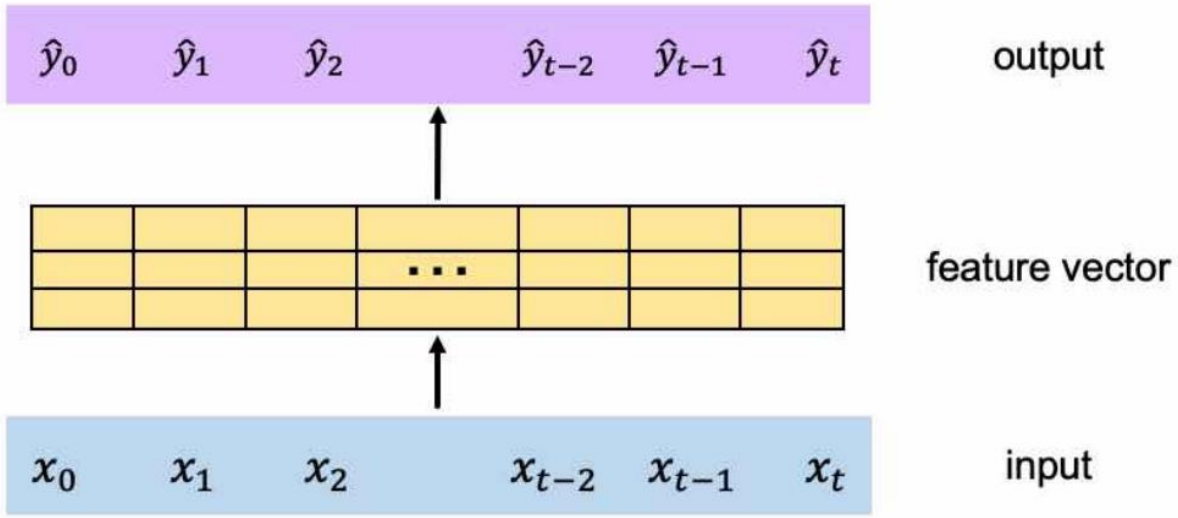
Goal of Sequence Modeling

Idea 1: Feed everything into dense network

- ✓ No recurrence
- ✗ Not scalable
- ✗ No order
- ✗ No long memory

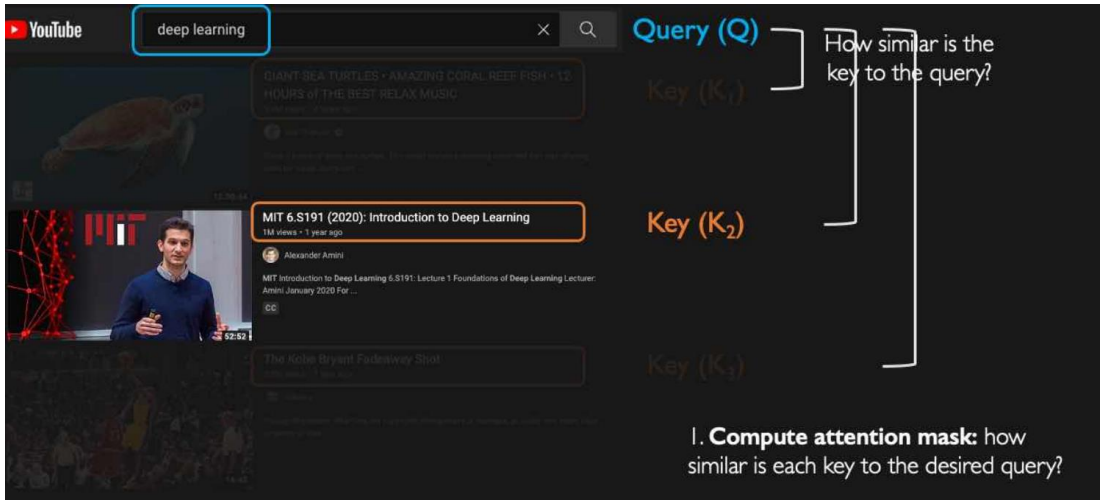
Idea: Identify and attend to what's important

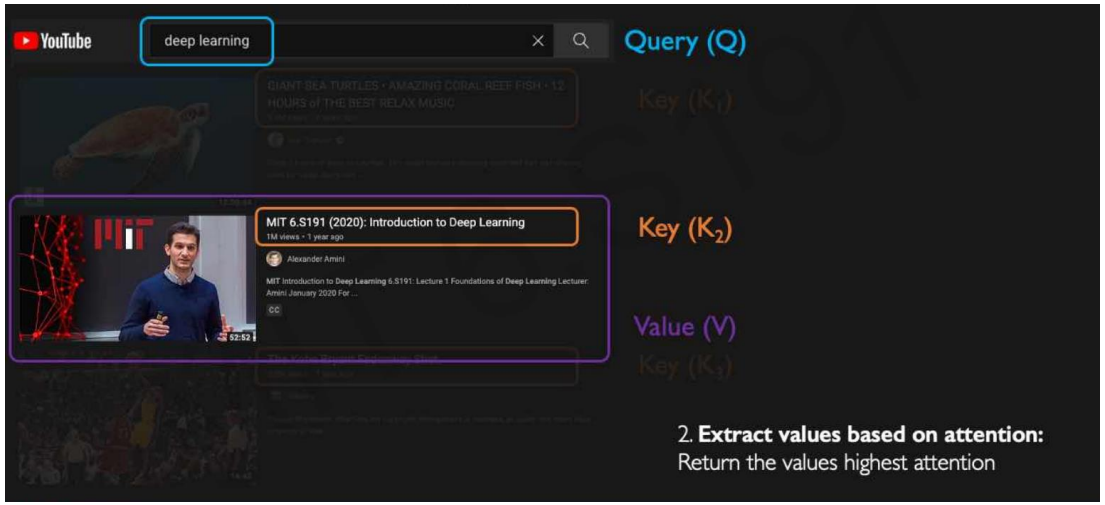
Can we eliminate the need for recurrence entirely?



출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

Understanding Attention with Search



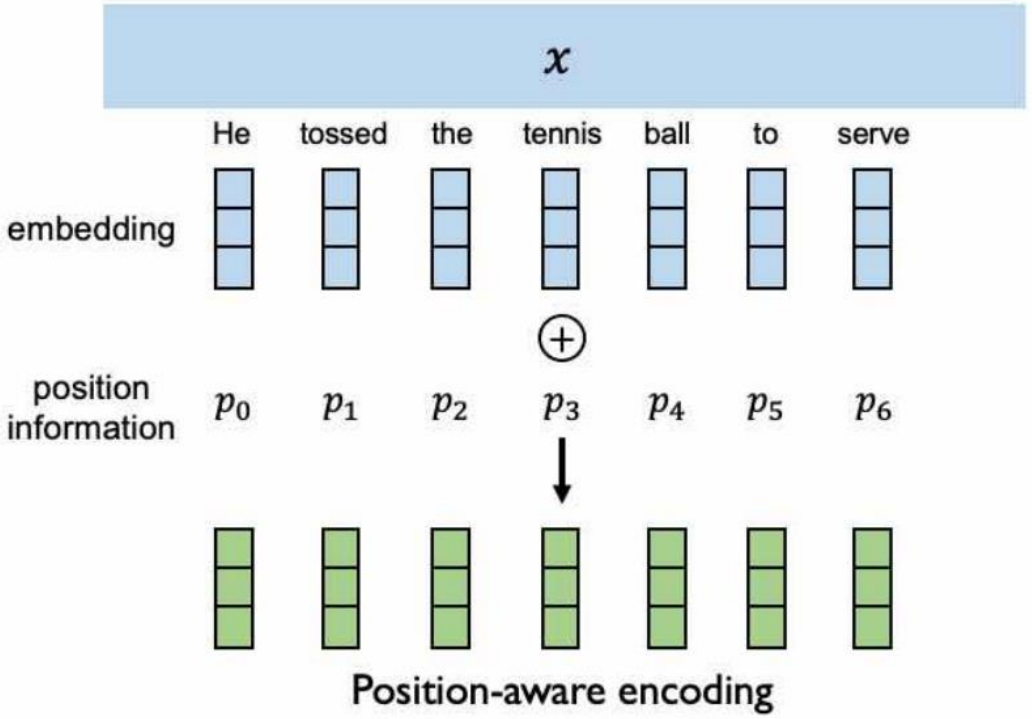


출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

Learning Self-Attention with Neural Networks

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

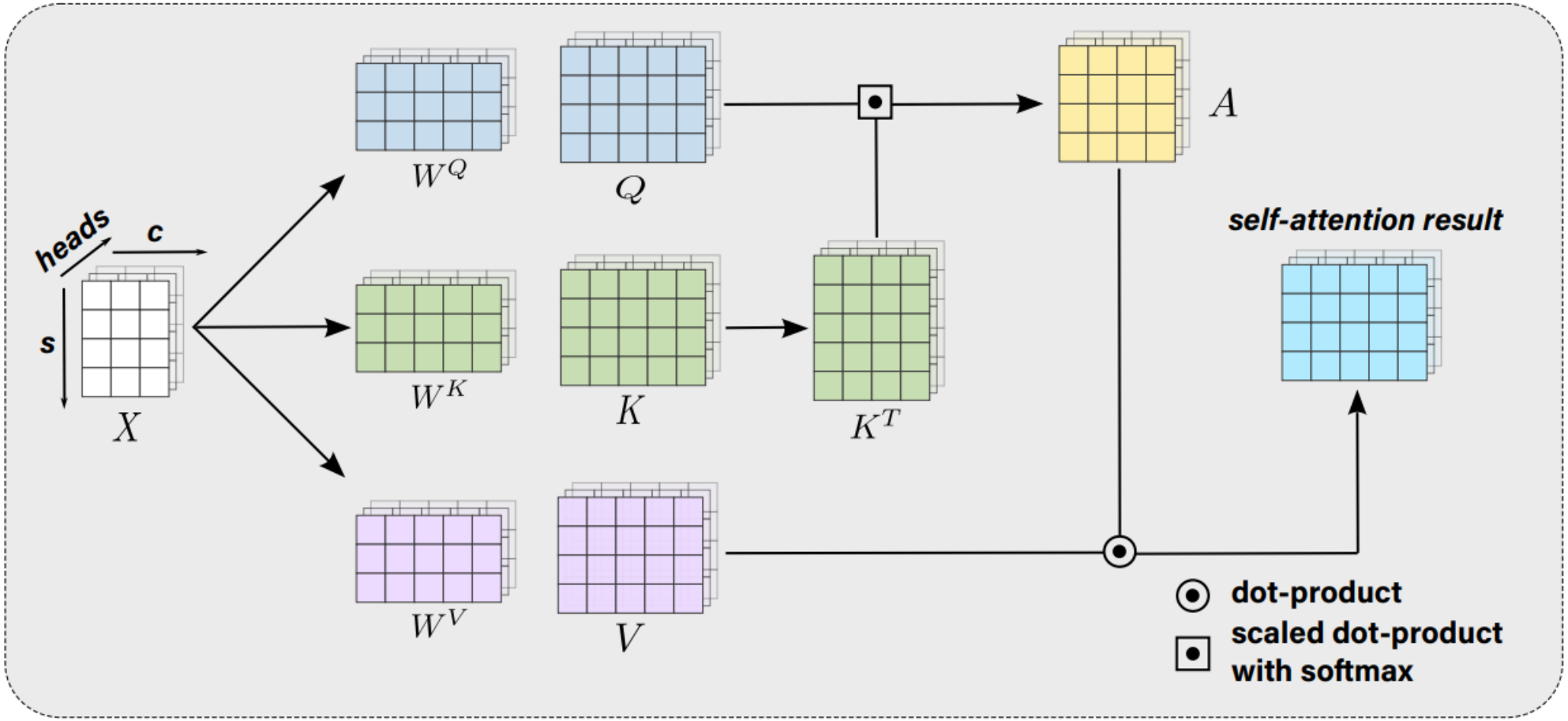
1. Encode **position** information
2. Extract query, key, value for search
3. Compute attention weighting
4. Extract features with high attention



Data is fed in all at once! Need to encode position information to understand order.

출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

Learning Self-Attention with Neural Networks



file:///C:/Users/admin/Downloads/Transformers_in_Medical_Image_Analysis__A_Review__Arxiv_v3_.pdf

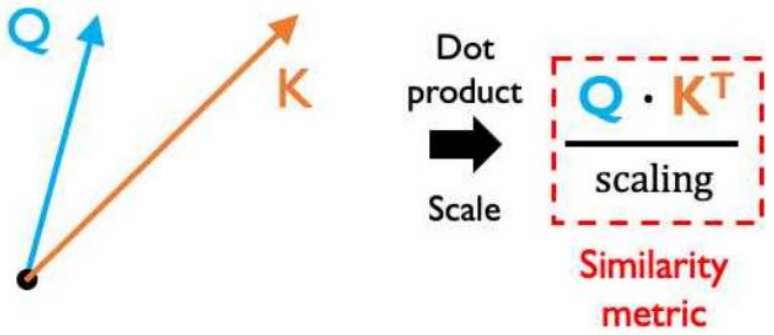
Learning Self-Attention with Neural Networks

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

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



Also known as the "cosine similarity"

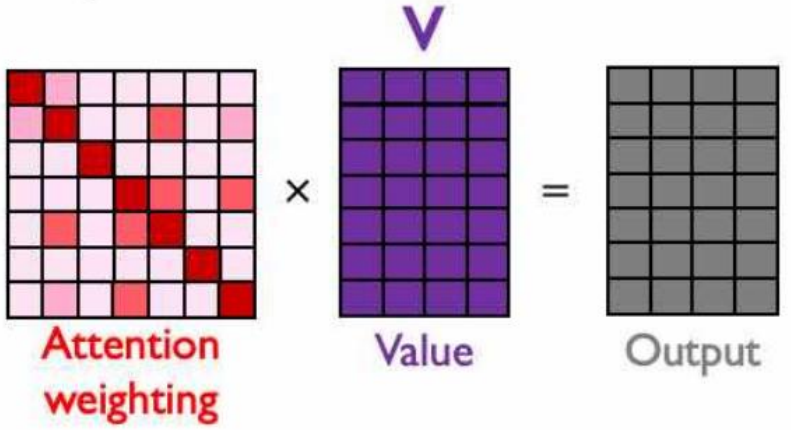
출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

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**

Last step: self-attend to extract features



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

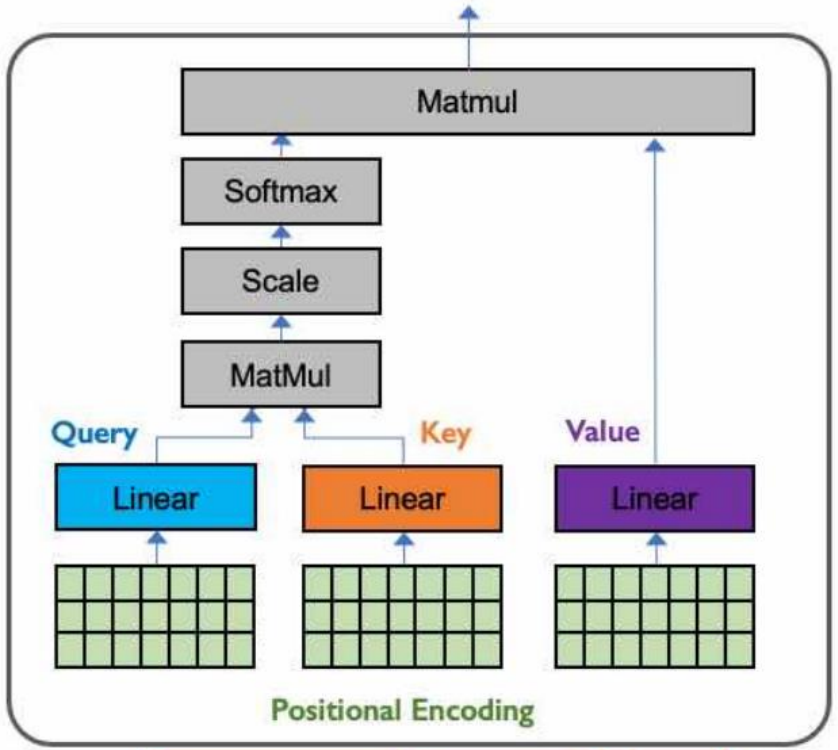
출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

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.



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

출처) http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

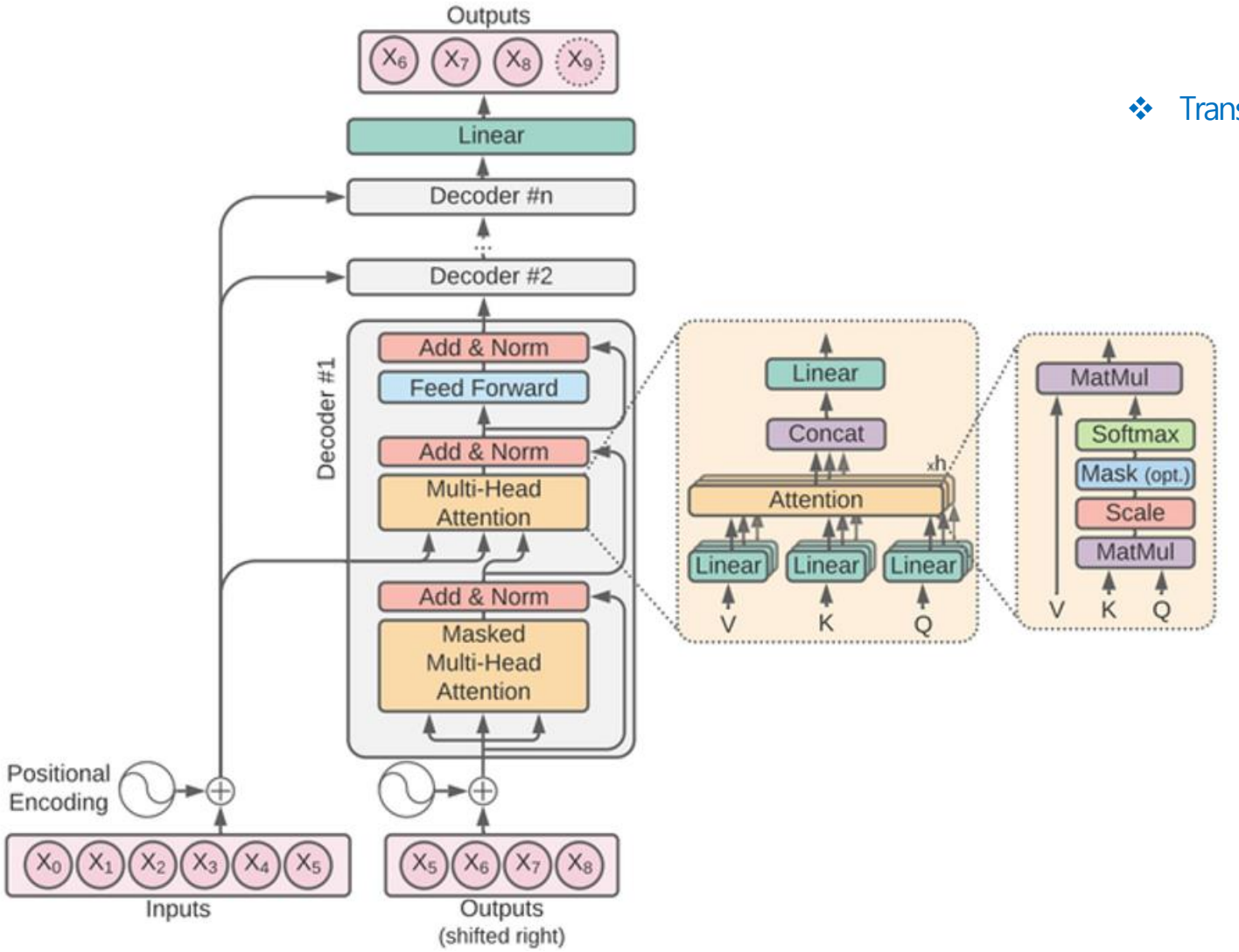
Comparing RNNs to Transformers

- RNNs
  - ✓ (+) LSTMs work reasonably well for long sequences.
  - ✓ (-) expects an ordered sequences of inputs
  - ✓ (-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.
- Transformer:
  - ✓ (+) Good at long sequences
    - Each attention calculation looks at all inputs
  - ✓ (+) Can operate over unordered sets or ordered sequences with positional encodings.
  - ✓ (+) Parallel computation
    - All alignment and attention scores for all inputs can be done in parallel
  - ✓ (-) Requires a lot of memory:
    - NxM alignment and attention scalers need to be calculated and stored for a single self-attention head

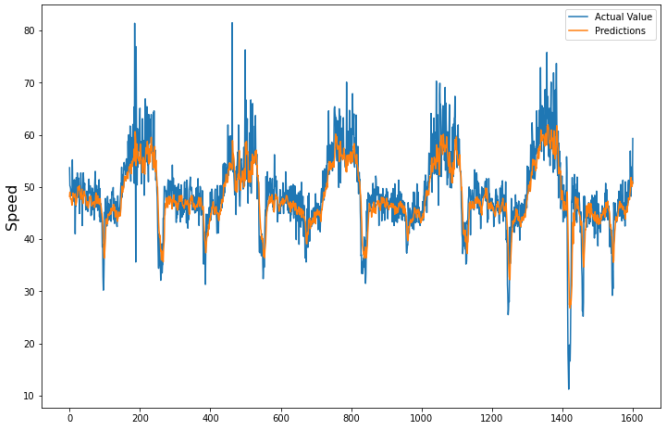
대상 도로 : 대학로 (유성구청네거리-구성삼거리)

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

멀티 헤드 어텐션 메커니즘



❖ TransformerVDS: Transformer+TimeEmbedding



요약

- 교통 데이터 VDS 분석 및 분류를 위한 라벨링
  - ✓ 교통량과 점유률 관계 및 상관계수
- DNN, RNN, LSTM, GRU로 교통 예측
  - ✓ RNN, LSTM, GRU 모델 및 가중치 공유
  - ✓ 파라미터 개수 구하기
  - ✓ Many-to-One, Many-to-Many 예측 문제 옵션 조정하기
- 1차원 CNN 모델로 교통 예측
  - ✓ 가중치 공유 알고리즘
  - ✓ 입력 데이터 형태(숫자, 텍스트, 이미지)에 따른 입력 변화
- 멀테헤드 어텐션과 트랜스포머 모델로 교통 예측하기
  - ✓ RNN, LSTM의 한계
  - ✓ Encoder-Decoder 모델
  - ✓ 쿼리, 키, 벨류 (Q,V,K) 기반 어텐션 메커니즘 이해
- 프로젝트
  - ✓ Multi-Step Ahead Prediction model에 적용하기

실습코드 : day1
vds01

Pandas를 이용하여 데이터 불러오기

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/vds16.csv")
```

```
df.head()
```

```
df.info()
```

```
df.describe()
```

```
plt.plot(df["ToVol"])
plt.show()
```

```
plt.plot(df["Date"][288:577],df["ToVol"][288:577],label='ToVol(2017.04.03.Monday)')
#plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.axvline(84, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(108, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(204, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(228, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)

plt.axhline(150, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.axhline(120, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.xticks(np.arange(0,288,24),rotation=45)
#plt.xticks((0, 24, 48, 72, 96, 120, 144, 168, 196, 216, 240, 264, 288), rotation=45)
plt.ylabel('Total Traffic Volume')
plt.xlabel('Date')
plt.legend()
plt.show()
```

vds01_label_visualization.ipynb

```
plt.plot(df["Date"][288:577],df["Speed"][288:577],label='Speed(2017.04.03.Monday)')
#plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.axvline(84,  0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(108, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(204, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(228, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)

plt.axhline(45, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.axhline(50, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.xticks(np.arange(0,288,24),rotation=45)
#plt.xticks((0, 24, 48, 72, 96, 120, 144, 168, 196, 216, 240, 264, 288), rotation=45)
plt.ylabel('Average Vehicle Speed')
plt.xlabel('Date')
plt.legend()
plt.show()
```

```
plt.plot(df["Date"][288:577],df["OccRate"][288:577],label='OccRate(2017.04.03.Monday)')
#plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.axvline(84,  0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(108, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(204, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)
plt.axvline(228, 0.04, 0.5, color='violet', linestyle='--', linewidth=2)

plt.axhline(5, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.axhline(8, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.xticks(np.arange(0,288,24),rotation=45)
#plt.xticks((0, 24, 48, 72, 96, 120, 144, 168, 196, 216, 240, 264, 288), rotation=45)
plt.ylabel('Occupancy Rate')
plt.xlabel('Date')
plt.legend()
plt.show()
```

```
plt.plot(df["Date"][:576],df["ToVol"][:576],label='ToVol')
plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.axvline(360, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(396, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(480, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(516, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axhline(150, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.axhline(120, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.xticks((0, 144, 288, 432,575), rotation=45)
plt.ylabel('Total Traffic Volume')
plt.xlabel('Date')
plt.legend()
plt.show()
```

```
plt.plot(df["Date"][:576],df["ToVol"][:576],label='ToVol')
plt.plot(df["Date"][:576],df["Speed"][:576]*1,label='Speed*2')
plt.plot(df["Date"][:576],df["OccRate"][:576]*5,label='OccRate*5')
plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.xticks((0, 144, 288, 432,575), rotation=45)
plt.xlabel('Date')
plt.legend()
plt.ylabel('ToVol, Speed, OccRate')
plt.show()
```

```
plt.plot(df["OccRate"],df["ToVol"],label='ToVol-OccRate')
plt.ylabel('Total Traffic Volume')
plt.xlabel('Occupancy Rate')
plt.legend()
plt.show()
```

vds01_label_visualization.ipynb

```
plt.plot(df["Date"][:576],df["ToVol"][:576],label='ToVol')
plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.axvline(360, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(396, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(480, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axvline(516, 0.4, 0.95, color='violet', linestyle='--', linewidth=2)
plt.axhline(150, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.axhline(120, 0.02, 0.99, color='limegreen', linestyle='--', linewidth=1)
plt.xticks((0, 144, 288, 432,575), rotation=45)
plt.ylabel('Total Traffic Volume')
plt.xlabel('Date')
plt.legend()
plt.show()
```

```
plt.plot(df["Date"][:576],df["ToVol"][:576],label='ToVol')
plt.plot(df["Date"][:576],df["Speed"][:576]*1,label='Speed*2')
plt.plot(df["Date"][:576],df["OccRate"][:576]*5,label='OccRate*5')
plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.xticks((0, 144, 288, 432,575), rotation=45)
plt.xlabel('Date')
plt.legend()
plt.ylabel('ToVol, Speed, OccRate')
plt.show()
```

```
plt.plot(df["OccRate"],df["ToVol"],label='ToVol-OccRate')
plt.ylabel('Total Traffic Volume')
plt.xlabel('Occupancy Rate')
plt.legend()
plt.show()
```

VDS 데이터 라벨링 전략

```
num_classes = 3
class_labels= ['Jam', 'Slow', 'Normal']
```

```
def get_score(speed):
    if speed < 45:
        label = 'Jam'
    elif speed < 50:
        label = 'Slow'
    else :
        label = 'Normal'
    return label
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

```
label = df['label'].unique()
label
```

Seaborn을 이용한 상관관계 등 가시화

```
sns.scatterplot(data=df, x = 'Date', y = 'Speed', hue='label')
```

```
sns.scatterplot(data=df[:576], x = 'Date', y = 'Speed', hue='label')
```

```
sns.scatterplot(data=df[:576], x = 'Date', y = 'ToVol', hue='label')
```

```
sns.scatterplot(data=df[:576], x = 'Date', y = 'OccRate', hue='label')
```

```
df.hist('ToVol',bins=100)
```

```
sns.heatmap(df.corr(), annot = True,  cmap='YlGnBu')
```

```
sns.countplot(x="label", data=df)
df.loc[:,'label'].value_counts()
```

```
sns.pairplot(df, hue='label')
```

```
sns.lmplot(x="ToVol", y="Speed", hue="label", data=df)
```

```
sns.lmplot(x="Speed", y="OccRate", hue="label", data=df)
```

```
sns.lmplot(x="OccRate", y="ToVol", hue="label", data=df)
```

```
sns.violinplot(x='label',y='Speed',data=df)
```

```
sns.violinplot(x='label',y='OccRate',data=df)
```

```
sns.violinplot(x='label',y='ToVol',data=df)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/vds16.csv")
```

```
df.head()
```

VDS 데이터 라벨링 전략

```
num_classes = 3
class_labels= ['Jam', 'Slow', 'Normal']
```

```
def get_score(speed):
    if speed < 45:
        label = 'Jam'
    elif speed < 50:
        label = 'Slow'
    else :
        label = 'Normal'
    return label
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

```
label = df['label'].unique()
label
```

```
sns.countplot(x="label", data=df)
```

```
sns.lmplot(x="OccRate", y="ToVol", hue="label", data=df)
```

```
X = df[['ToVol','LaVol','MeVol','SmVol', 'Speed','OccRate']]
y = df['label']
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_scaled[:5]
```

```
X1=df[['OccRate','ToVol']].values

X2=df[['Speed','ToVol']].values
```

Train/Test 데이터 나누기

```
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report,  accuracy_score, confusion_matrix

from mlxtend.plotting import plot_decision_regions
#!pip install mlxtend
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=0)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
X_train
```

vds02_machinelearning.ipynb

```
# LogisticRegression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
# Training a classifier
lr = LogisticRegression()
lr.fit(X1, y)
plot_decision_regions(X1, y, clf=lr, legend=2)

# Adding axes annotations
plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('Loginstic Regression on VDS')
plt.show()
```

```
Model = SVC()
Model.fit(X_train, y_train)
y_pred = Model.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
svm = SVC()
svm.fit(X1, y)
plot_decision_regions(X1, y, clf=lr, legend=2)

plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('SVM on VDS')
plt.show()
```

```
svm = SVC()
svm.fit(X2, y)
plot_decision_regions(X2, y, clf=lr, legend=2)
plt.xlabel('Speed')
plt.ylabel('Traffic Volume')
plt.title('SVM on VDS')
plt.show()
```

```
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
# Training a classifier
tree = DecisionTreeClassifier()
tree.fit(X1, y)
plot_decision_regions(X1, y, clf=tree, legend=2)

# Adding axes annotations
plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('Decision Tree on VDS')
plt.show()
```

```
Model=RandomForestClassifier(max_depth=2)
Model.fit(X_train,y_train)
y_pred=Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_pred,y_test))
#Accuracy Score
print('accuracy is ',accuracy_score(y_pred,y_test))
```

```
forest = RandomForestClassifier(max_depth=2)
forest.fit(X1, y)
plot_decision_regions(X1, y, clf=forest, legend=2)


# Adding axes annotations
plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('Random Forest on VDS')
plt.show()
```

```
# K-Nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier

Model = KNeighborsClassifier(n_neighbors=8)
Model.fit(X_train, y_train)
y_pred = Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
# Training a classifier
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X1, y)
plot_decision_regions(X1, y, clf=knn, legend=2)

# Adding axes annotations
plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('KNN on VDS')
plt.show()
```

```
from sklearn.ensemble import GradientBoostingClassifier
Model=GradientBoostingClassifier()
Model.fit(X_train,y_train)
y_pred=Model.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_pred,y_test))

#Accuracy Score
print('accuracy is ',accuracy_score(y_pred,y_test))
```

```
# Training a classifier
gboost=GradientBoostingClassifier()
gboost.fit(X1, y)
plot_decision_regions(X1, y, clf=gboost, legend=2)

# Adding axes annotations
plt.xlabel('OccRate')
plt.ylabel('Traffic Volume')
plt.title('GradientBoosting on VDS')
plt.show()
```

필요한 라이블러리 불러오기

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,  accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder, StandardScaler

from mlxtend.plotting import plot_decision_regions
```

```
df = pd.read_csv("https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/vds16.csv")
```

```
df.head()
```

```
plt.plot(df["Date"][:576],df["ToVol"][:576],label='ToVol')
plt.plot(df["Date"][:576],df["Speed"][:576]*1,label='Speed*2')
plt.plot(df["Date"][:576],df["OccRate"][:576]*5,label='OccRate*5')
plt.axvline(288, 0.02, 0.99, color='lightgray', linestyle='--', linewidth=2)
plt.xticks((0, 144, 288, 432,575), rotation=45)
plt.xlabel('Date')
plt.legend()
plt.show()
```

Types of Species

```
num_classes = 3
class_labels= ['Jam', 'Slow', 'Normal']
```

```
def get_score(speed):
    if speed < 45:
        label = 'Jam'
    elif speed < 50:
        label = 'Slow'
    else :
        label = 'Normal'
    return label
```

```
df["label"] = df["Speed"].apply(lambda spd: get_score(spd))
df.head()
```

```
label = df['label'].unique()
label
```

vds03_dnn_classification.ipynb

```
X = df[['ToVol','LaVol','MeVol','SmVol', 'Speed','OccRate']]
y = df['label']
```

```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

Train/Test 데이터 나누기

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                                    test_size=0.3, random_state=1)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
X_train[:2]
```

```
#import keras
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

from livelossplot import PlotLossesKeras
```

```
def model_shallow():
    model = Sequential([
        Dense(16, input_dim=X_train.shape[1], activation = 'relu'),
        Dense(num_classes, activation= 'softmax')
    ])
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

```
model = model_shallow()
#history = model.fit(X_train,y_train, epochs=100, validation_split=0.2)
history = model.fit(X_train,y_train, epochs=100, validation_split=0.2,callbacks=[PlotLossesKeras()])
```

```
model.evaluate(X_test,y_test)
```

```
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
import numpy as np

y_pred_labels = [np.argmax(label) for label in y_pred]
cm = confusion_matrix(y_test,y_pred_labels)
sns.heatmap(cm , annot = True,  cmap='YlGnBu',
            fmt = 'd',xticklabels = class_labels,
            yticklabels = class_labels)
```

vds03_dnn_classification.ipynb

```
def model_deep():
    model = Sequential([
        Dense( 32, input_dim=X_train.shape[1], activation = 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( units = 32, activation= 'relu'),
        Dense( num_classes, activation= 'softmax')
    ])
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

```
model = model_deep()
history = model.fit(X_train,y_train, epochs=100, validation_split=0.2, callbacks=[PlotLossesKeras()])
```

```
model.evaluate(X_test, y_test)
```

```
from sklearn.metrics import confusion_matrix
import numpy as np
y_pred = model.predict(X_test)
y_pred_labels = [np.argmax(label) for label in y_pred]
cm = confusion_matrix(y_test,y_pred_labels)
#sns.heatmap(cm , annot = True, fmt = 'd',xticklabels = class_labels,yticklabels = class_labels)
sns.heatmap(cm , annot = True,  cmap='YlGnBu',
            fmt = 'd',xticklabels = class_labels, yticklabels = class_labels)
```

실습코드 : day2

Vds06_many2one_model (1/7)

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib as mpl
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, Flatten, Conv1D, GRU, MaxPool1D
from keras import metrics
from livelossplot import PlotLossesKeras
```

```
def plot_windows(X, Y, Y_pred):
    plt.figure(figsize=(8, 4))
    plt.plot(X[0,:], "o-")
    if Y is not None:
        plt.plot(np.arange(look_back, look_back + look_forward), Y[0, :], "go", label="Actual")
    if Y_pred is not None:
        plt.plot(np.arange(look_back, look_back + look_forward), Y_pred[0, :], "rx-", label="Forecast", markersize=10)
    plt.legend(fontsize=12)
    plt.grid(True)
    plt.axvline(look_back, color='gray',linestyle='--',linewidth=2)
    plt.show()
```

Vds06_many2one_model (2/7)

```
def plot_pred():
    plt.figure(figsize=(5, 4))
    plt.scatter(y_test.flatten(), y_pred.flatten())
    plt.xlabel('Acutal')
    plt.ylabel('Predictions [Sine Wave]')
    plt.plot([0,1], [0,1],color='gray',linestyle='--',linewidth=2)
    plt.show()
```

```
def plot_learning_curves(loss, val_loss):
    plt.figure(figsize=(8, 4))
    plt.plot(np.arange(len(loss)) + 0.5, loss, "b.-", label="Training loss")
    plt.plot(np.arange(len(val_loss)) + 1, val_loss, "r.-", label="Validation loss")
    plt.gca().xaxis.set_major_locator(mpl.ticker.MaxNLocator(integer=True))
    #plt.axis([1, 20, 0, 0.05])
    plt.legend(fontsize=14)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.grid(True)
```

Generate the Dataset

```
df = pd.read_csv('https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/vds16.csv')
df.head()
```

```
df['ToVol'][:576].plot(rot=45,figsize=(8,4))
```

```
#features = ['ToVol','LaVol','MeVol','SmVol','OccRate']
df = df[['ToVol']]
```

```
df
```

```
# target = Speed
# df_y = df.iloc[:,5:6].values
```

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
#scaler = StandardScaler()
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df)
#scaled_y = scaler.fit_transform(df_y)
```

```
print(df_scaled)
#print(scaled_y)
```

Vds06_many2one_model (3/7)

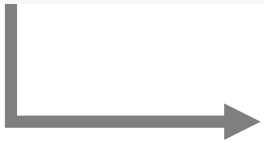
```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back, look_forward):
    dataX, dataY = [], []
    np.array(dataY)
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back)]
        if len(dataset[i + look_back:i+look_back+look_forward]) == look_forward:
            dataX.append(a)
            dataY.append(dataset[i + look_back:i+look_back+look_forward])
    return np.array(dataX), np.array(dataY)
```

```
look_back = 12*5
look_forward = 1
```

```
X, y = create_dataset(df_scaled,look_back,look_forward)
```

```
X.shape, y.shape
```

```
plot_windows(X,y,y)
```



```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, shuffle=False)
print('X_train:', X_train.shape)
print('X_test :', X_test.shape)
print('y_train:', y_train.shape)
print('y_test :', y_test.shape)
```

1) many-to-one with Baselines (MSE)

Naive predictions (just predict the last observed value):

```
y_pred = X_test[:, -1]
```

```
y_pred = y_pred[:,np.newaxis]
X_test.shape, y_test.shape, y_pred.shape
```

```
def model_mse():
    model = np.mean(tf.keras.losses.mean_squared_error(y_test, y_pred))
    return model
```

```
plot_windows(X_test[:, :], y_test[:,], y_pred[:,])
```

```
model = model_mse()
a1 = model
print('m1_mse: ', a1)
```

```
y_max =y_test.flatten()[:,].max()
y_min =y_test.flatten()[:,].min()
print(y_max,y_min)
plot_pred()
```

2) many-to-one with DNN

```
def model_dnn():
    model = Sequential([ Flatten(input_shape=[look_back,1]),
                         Dense(32, activation='relu'),
                         Dense(32, activation='relu'),
                         Dense(look_forward) ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

```
model = model_dnn()
history = model.fit(X_train, y_train, epochs=30,validation_split=0.2,  callbacks=[PlotLossesKeras()])
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
plt.show()
```

```
a2, _ = model.evaluate(X_test, y_test)
print('m2_dense: ', a2)
plot_pred()
```

Vds06_many2one_model (5/7)

3) many-to-one with SimpleRNN

```
def model_simpleRNN():
    model = Sequential([
        SimpleRNN(32, return_sequences=True, input_shape=[look_back, 1]),
        SimpleRNN(32,return_sequences=True),
        SimpleRNN(look_forward,return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

```
model = model_simpleRNN()
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
plot_pred()
a3, _ = model.evaluate(X_test, y_test)
print('m3_simpleRnn: ', a3)
```

4) many-to-one with LSTM

```
def model_lstm():
    model = Sequential([
        LSTM(32, return_sequences=True, input_shape=[look_back, 1]),
        LSTM(32, return_sequences=True),
        LSTM(look_forward, return_sequences=False)
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

```
model = model_lstm()
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
a4, _ = model.evaluate(X_test, y_test)
print('m1_LSTM: ', a4)
```

```
plot_pred()
```

Vds06_many2one_model (6/7)

many-to-one with GRU

```
def model_gru():
    model = Sequential([
        GRU(32, return_sequences=True, input_shape=[look_back, 1]),
        GRU(32, return_sequences=True),
        GRU(look_forward, return_sequences=False),
    ])
    model.compile(loss="mse", optimizer='adam',metrics =['mae'])
    model.summary()
    return model
```

```
model = model_gru()
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
a5, _ = model.evaluate(X_test, y_test)
print('m1_GRU: ', a5)
```

```
plot_pred()
```

6) many-to-one with CNN1D

```
def model_conv1d():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid', activation='relu',
               input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        Flatten(),
        Dense(32, activation='relu'),
        Dense(look_forward,  activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```

```
model = model_conv1d()
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
a6, _ = model.evaluate(X_test, y_test)
print('m1_Conv1D: ', a6)
```

Vds06_many2one_model (7/7)

7) many-to-one with LSTM-CNN1D

```
def model_cnnlstm():
    model = Sequential([
        Conv1D(filters=32, kernel_size=3, strides=1, padding='valid', activation='relu',
               input_shape = (look_back,1)),
        MaxPool1D(2, padding = 'valid'),
        LSTM(32, return_sequences=True),
        Flatten(),
        Dense(32),
        Dense(look_forward, activation='relu')
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```

```
model = model_cnnlstm()
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
a7, _ = model.evaluate(X_test, y_test)
print('m1_Hybrid: ', a7)
```

```
df = pd.DataFrame({
    'models': ['mse', 'Dense', 'SimpleRNN','LSTM','GRU', 'Conv1D','CNN-LSTM' ],
    'loss':[a1, a2, a3, a4, a5, a6, a7]})
df=df.sort_values(by='loss', ascending=True)
```

```
df
```

```
df.plot.bar(x='models',y='loss',rot=15)
```

실습코드 : day3

PART II: Forecasting Several Steps Ahead

```
look_back = 12*5
look_forward = 12*24
```

```
X, y = create_dataset(df_scaled,look_back,look_forward)
```

```
plot_windows(X,y,y)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20, shuffle=False)

print('X_train:', X_train.shape)
print('X_test :', X_test.shape)
print('y_train:', y_train.shape)
print('y_test :', y_test.shape)
```

(1) m2: many-to-many with DNN

```
model = model_dnn()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
b1, _ = model.evaluate(X_test, y_test)
print('m2_SimpleRNN): ', b1)
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
plot_pred()
```

model (b2) : Deep SimpleRNN

- Now let's create an RNN that predicts all 10 next values at once:

```
model = model_simpleRNN()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
plot_pred()
```

```
b2, _ = model.evaluate(X_test, y_test)
print('m2_SimpleRnn): ', b2)
```

b3) many to many with LSTM

```
model = model_lstm()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
b3, _ = model.evaluate(X_test, y_test)
print('horizon model b3: ', b3)
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
plot_pred()
```

Vds07_many2many_model (3/3)

b4) many to many with GRU

```
model =  model_gru()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
b4, _ = model.evaluate(X_test, y_test)
print('horizon model b4[LSTM]: ', b4)
```

```
y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])
```

```
plot_learning_curves(history.history["loss"], history.history["val_loss"])
```

```
plot_pred()
```

(b5) many to many with Conv1D

```
model = model_conv1d()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
b5, _ = model.evaluate(X_test, y_test)
print('horizon model b5[GRU]: ', b5)

y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])

plot_learning_curves(history.history["loss"], history.history["val_loss"])

plot_pred()
```

(b6) many to many with hybrid model : Conv1D+LSTM

```
model =model_cnnlstm()
history = model.fit(X_train, y_train, epochs=30, validation_split=0.2, verbose=1)
```

```
b6, _ = model.evaluate(X_test, y_test)
print('horizon model b6[Conv1D]: ', b6)

y_pred = model.predict(X_test)
plot_windows(X_test[:, :], y_test[:, :], y_pred[:, :])

plot_learning_curves(history.history["loss"], history.history["val_loss"])

plot_pred()
```

Summary of Many to Many Model

```
df = pd.DataFrame({
    'models': ['DNN', 'SimpleRNN','lSTM','GRU', 'Conv1D','ConvLSTM'],
    'loss':[b1, b2, b3, b4, b5, b6]})
df=df.sort_values(by='loss', ascending=True)
```

```
df
```

```
df.plot.bar(x='models', y='loss', rot=0)
```

KISTI SINCE 1962

2022

TRUST KISTI

Korea Institute of Science
and Technology Information