# 상황 정보를 이용한 딥러닝 기반 전력수요예측

2018. 11. 07

Advisor : Professor 최원익

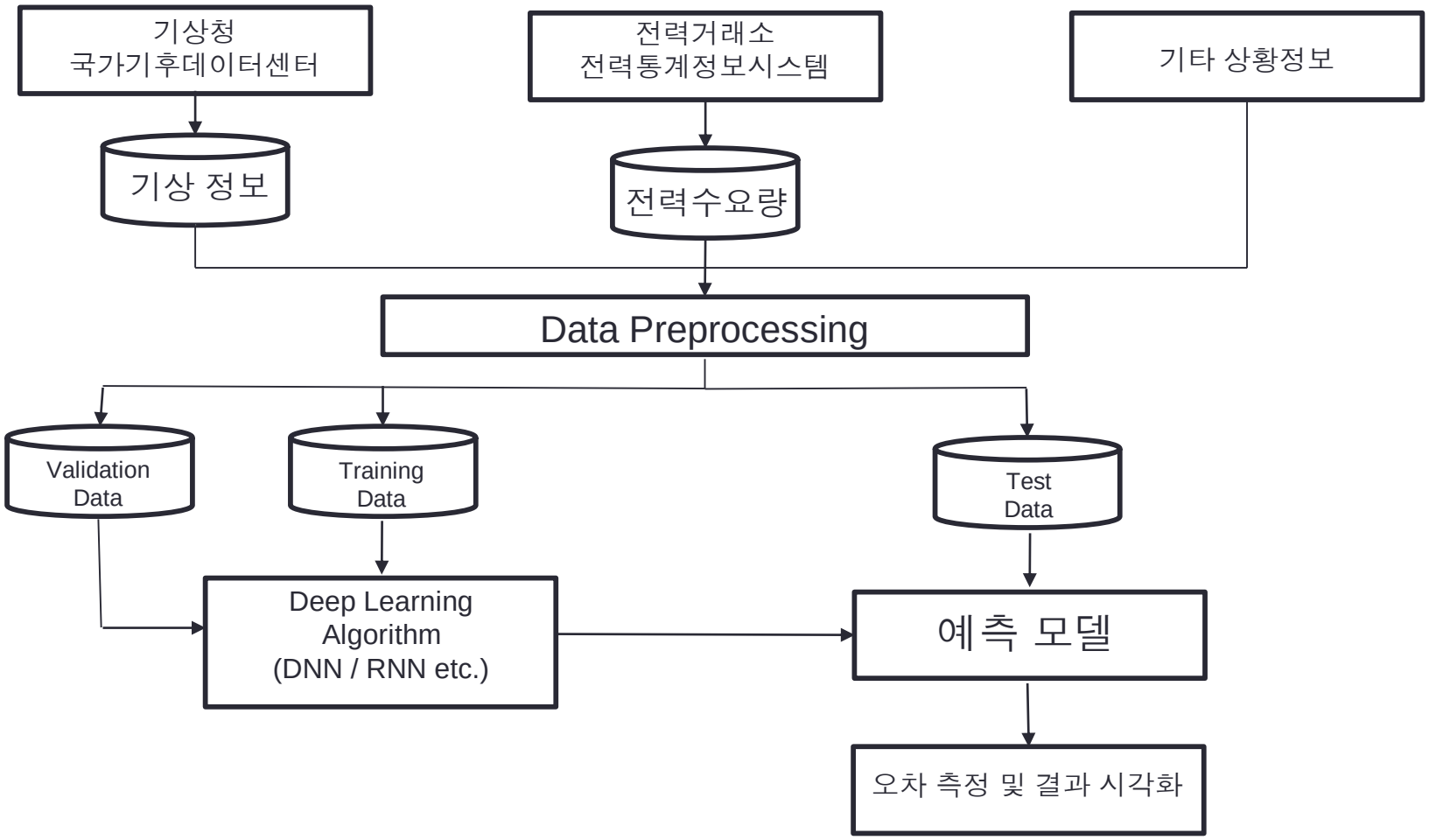
전승주

Department of Information and Communication Engineering,
Inha University

# 목차

1. 설계블록도
2. 데이터 선정 · 수집 · 가공
3. DNN 모델
4. RNN-LSTM 모델
5. 성능평가
6. 향후 계획

# 설계블록도


기상청
국가기후데이터센터
전력거래소
전력통계정보시스템
기타 상황정보
기상 정보
전력수요량
Data Preprocessing
Validation Data
Training Data
Test Data
Deep Learning Algorithm (DNN / RNN etc.)
예측 모델
오차 측정 및 결과 시각화

# 데이터 선정 · 수집 · 가공

- 데이터 선정

1) 시간정보
   월, 일, 요일, 시간, 휴일여부

2) 기상정보
   기온, 풍속, 습도, 증기압, 일사량

3) 과거 전력 수요
   일별 최대전력, 시간별 현재수요

# 데이터 선정 · 수집 · 가공

- 데이터 수집

1) 시간정보
일별 : 2006.01.01~2018.09.30 (4642일)
시간별 : 2012.06.01.00~2018.09.30.23 (55513시간)

2) 기상정보
기상자료개방포털 : https://data.kma.go.kr/data/
지역별 : 서울, 대전, 대구, 광주, 부산
기온, 풍속, 습도, 증기압, 일사량

3) 과거 전력 수요
전력통계정보시스템 : http://epsis.kpx.or.kr/epsisnew/
일별 최대전력
시간별 현재전력수요

# 데이터 선정 · 수집 · 가공

- 데이터 가공

1) 빈 데이터

수집한 데이터의 빈 값을 채우기 위해 전·후 1개의 데이터(총 2개의 데이터)의 평균값으로 대체

2) 대표 기상정보

전국 단위의 대표 기상정보로 변환하기 위한 가중치

| 지역 | 서울 | 대전 | 대구 | 광주 | 부산 |
|---|---|---|---|---|---|
| 가중치(%) | 50.0 | 10.0 | 12.0 | 8.0 | 20.0 |

출처 : 신이레, 윤상후 (2016). 전력수요예측을 위한 기상정보 활용성평가

# 데이터 선정 · 수집 · 가공

- 데이터 가공

3) normalization

① 주기성을 갖는 시간정보

월, 일, 요일, 시간 정보는 특정 범위에서 값이 반복되는 **주기성**을 갖는다.
0~1값으로 만들기 위해 단순히 max값으로 나누는 방식은 주기의 경계에서 큰 값의 변화를 가져온다. 즉 주기성을 반영하지 못한다.

예를 들어, 일 정보를 $\frac{1}{30}, \frac{2}{30,}, \ldots, \frac{30}{30}$ 와 같이 바꾸면 $\frac{30}{30}$에서 $\frac{1}{30}$로 바뀔 때 실제로 하루 차이이지만 큰 값의 변화가 생긴다.

# 데이터 선정 · 수집 · 가공

- 데이터 가공

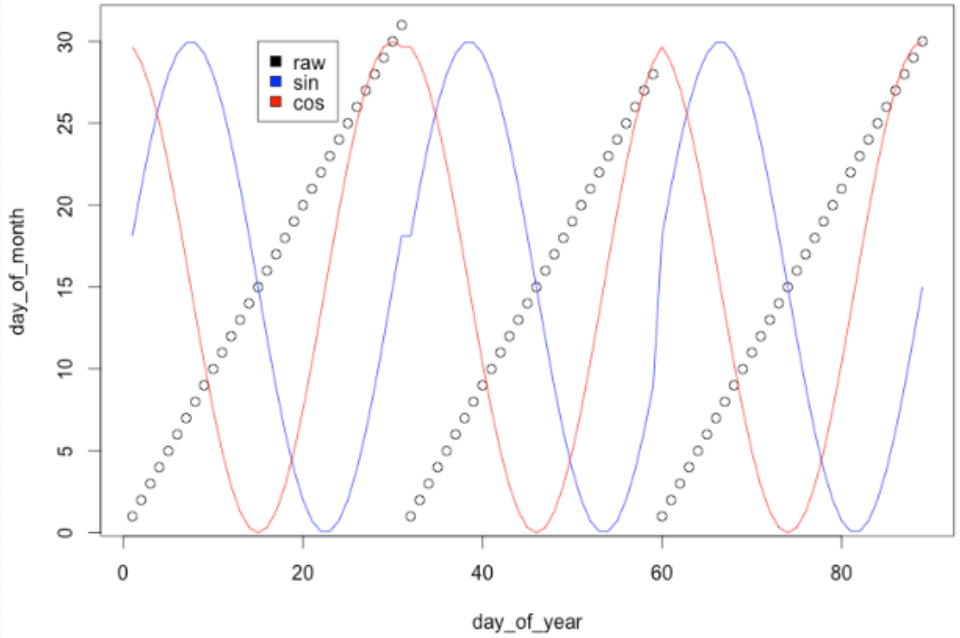
3) normalization

① 주기성을 갖는 시간정보

시간정보의 주기성을 반영하기 위해 sin, cos함수를 이용하였다.

$$y = 0.25\sin\frac{2\pi}{30}x + 0.25\cos\frac{2\pi}{30}x$$

일 정보의 한 달 길이를 고려하여
수식에 반영하였고,
월, 요일, 시간 정보에 대하여도
마찬가지 방식으로 값을 변환하였다.




출처 : https://stats.stackexchange.com/questions/126230/optimal-construction-of-day-feature-in-neural-networks

# 데이터 선정 · 수집 · 가공

- 데이터 가공

3) normalization

② 기상정보와 전력수요정보

feature scaling의 대표적인 2가지 방식

min-max scaling vs standardization

$$\frac{x - min}{\max - min} \qquad \frac{x - mean}{stddev}$$

샘플데이터에 대한 성능 평가 후 min-max scaling 방식을 선택

최종데이터 형태

| 월 | 일 | 시 | 요일 | 휴일여 | 기온(℃) | 풍속(m/s) | 습도(%) | 증기압(hPa) | 일사(MJ/m2) | 현재수요(MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.25 | 0.796515 | 0.75 | 0.200638 | 0 | 0.629707 | 0.173596672 | 0.6714450859 | 0.4189181153 | 0 | 0.26973344918 |
| 0.25 | 0.796515 | 0.806186 | 0.200638 | 0 | 0.619403 | 0.125155924 | 0.7005780339 | 0.4180868107 | 0 | 0.23988548342 |
| 0.25 | 0.796515 | 0.841506 | 0.200638 | 0 | 0.611751 | 0.123700622 | 0.7271676292 | 0.4215307867 | 0 | 0.24028198656 |
| 0.25 | 0.796515 | 0.853553 | 0.200638 | 0 | 0.608796 | 0.150103949 | 0.7424277448 | 0.4244403526 | 0 | 0.23354161287 |
| 0.25 | 0.796515 | 0.841506 | 0.200638 | 0 | 0.599856 | 0.152598751 | 0.7595375714 | 0.4192150097 | 0 | 0.2381178526 |
| 0.25 | 0.796515 | 0.806186 | 0.200638 | 0 | 0.593454 | 0.102702702 | 0.7902890164 | 0.4246184893 | 0 | 0.24545810589 |
| 0.25 | 0.796515 | 0.75 | 0.200638 | 0 | 0.593189 | 0.189604988 | 0.778265895 | 0.4180274318 | 0.013732193341 | 0.25362607074 |
| 0.25 | 0.796515 | 0.676777 | 0.200638 | 0 | 0.607167 | 0.162785861 | 0.722080924 | 0.4117332688 | 0.071794869749 | 0.27764058509 |
| 0.25 | 0.796515 | 0.591506 | 0.200638 | 0 | 0.635162 | 0.112889812 | 0.6605780339 | 0.4210557555 | 0.21150996548 | 0.30165509944 |

# DNN 모델

1) 일별

① Input & Output data

| Parameter | Explanation | Number |
|---|---|---|
| $y$ | 예측일 전력수요 예측 값 | 1 |
| $x_1, x_2, \dots, x_n$ | 과거 n일동안 전력수요 | n |
| $x_{n+1}, \dots, x_{n+5}$ | 예측 전일 기상정보<br>(기온, 풍속, 습도, 증기압, 일사량) | 5 |
| $x_{n+6}, \dots, x_{n+9}$ | 예측일 월, 일, 요일, 휴일여부 | 4 |

# DNN 모델

1) 일별

① Input & Output data

· data set 구성 코드

```
train_set = np.loadtxt('C:\exercise_data\\train_value_min_max_scalar4_.csv', delimiter=',', usecols=range(10))
test_set = np.loadtxt('C:\exercise_data\\test_value_min_max_value4_.csv', delimiter=',', usecols=range(10))
train_dataX = []
train_dataY = []
test_dataX = []
test_dataY = []


i = past_load_num
num = len(train_set)
while i < num:
    if train_set[i][9]==0:
        i+=1
        continue

    tempX = []
    tempY = []
    for j in range(9):
        if j>=0 and j<4:
            tempX.append(train_set[i][j])
        else:
            tempX.append(train_set[i-1][j])

    for k in range(past_load_num):
        tempX.append(train_set[i-past_load_num+k][9])

    train_dataX.append(tempX)
    tempY.append(train_set[i][9])
    train_dataY.append(tempY)
    print(i)
    i+=1
```

시간정보 $(x_{n+6}, \dots, x_{n+9})$

기상정보 $(x_{n+1}, \dots, x_{n+5})$

과거전력수요 $(x_1, x_2, \dots, x_n)$

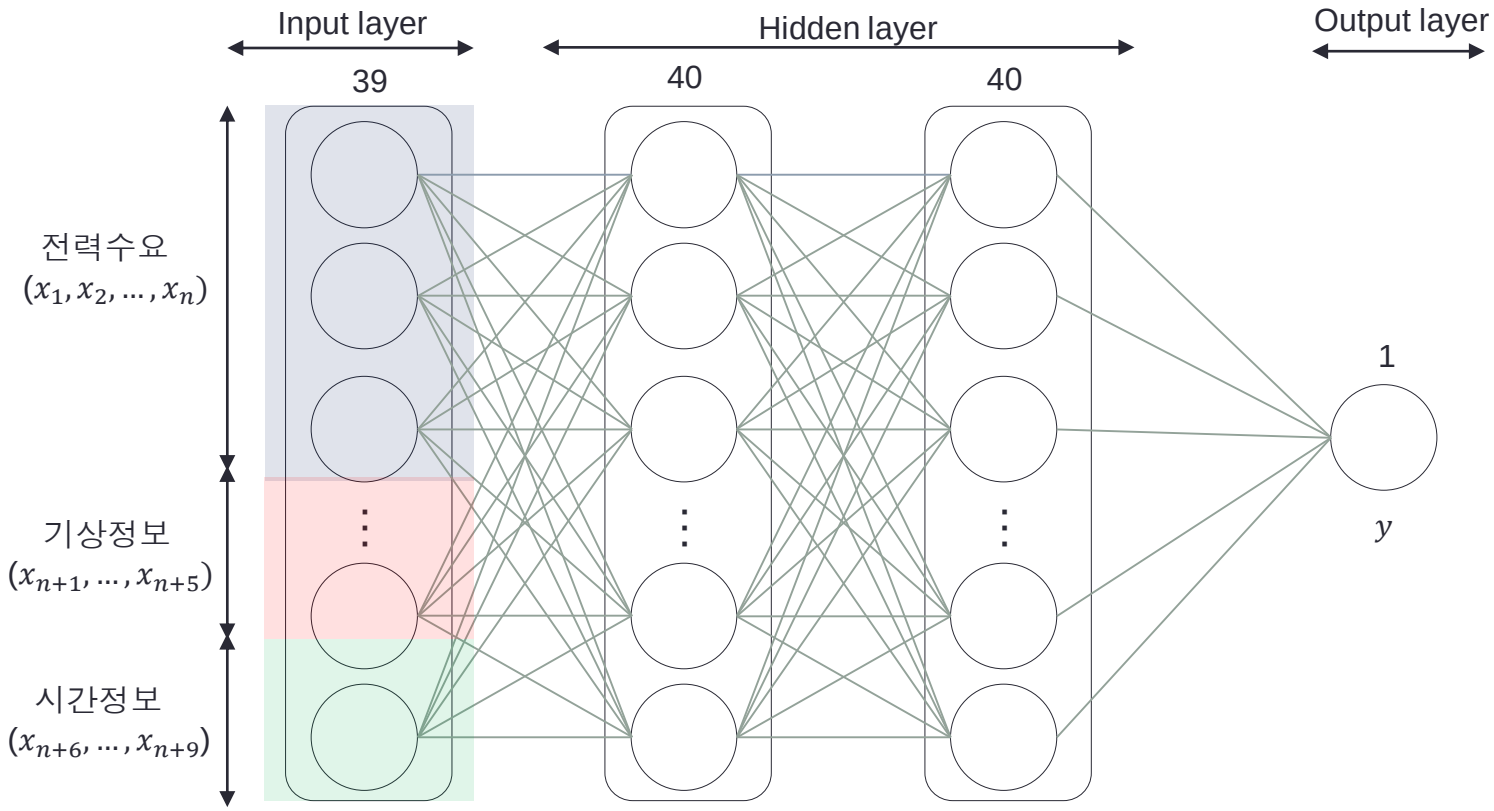
예측전력수요 (label, $y$)

n=30
trainX : 3621 X 39
trainY : 3621 X 1
testX : 959 X 39
testY : 959 X 1

# DNN 모델

1) 일별

② Neural Net 구조 (n=30)



# DNN 모델

2) 시간별

① Input & Output data

| Parameter | Explanation | Number |
|---|---|---|
| $y_1, y_2, \dots, y_{24}$ | 24시간 전력수요 예측 값 | 24 |
| $x_1, x_2, \dots, x_{24n}$ | 과거 n일동안 시간별 전력수요 | 24n |
| $x_{24n+1}, \dots, x_{24n+120}$ | 예측 전일 시간별 기상정보<br>(기온, 풍속, 습도, 증기압, 일사량) | 120 |
| $x_{24n+121}, \dots, x_{24n+125}$ | 예측일 월, 일, 요일, 시간, 휴일여부 | 5 |

# DNN 모델

2) 시간별

① Input & Output data

· data set 구성 코드

```
train_set = np.loadtxt('C:\exercise_data\\original_hours_4_train.csv', delimiter=',', usecols=range(12))
test_set = np.loadtxt('C:\exercise_data\\original_hours_4_test.csv', delimiter=',', usecols=range(12))
train_dataX = []
train_dataY = []
test_dataX = []
test_dataY = []


i = past_load_num
num = len(train_set)
while i <= num-24:
    for t in range(24):
        if train_set[i+t][11]==0:
            i+=24
            continue

    tempX = []
    tempY = []
    for n in range(24):
        for j in range(11):
            if j==0:
                continue
            elif n==0 and j<=5:
                tempX.append(train_set[i+n][j])
            elif j>=6:
                tempX.append(train_set[i-24+n][j])


    for k in range(past_load_num):
        tempX.append(train_set[i-past_load_num+k][11])

    train_dataX.append(tempX)

    for m in range(24):
        tempY.append(train_set[i+m][11])
    train_dataY.append(tempY)
    #print(i)
    i+=24
```

시간정보 $(x_{24n+121}, \dots, x_{24n+125})$

기상정보 $(x_{24n+1}, \dots, x_{24n+120})$

과거전력수요 $(x_1, x_2, \dots, x_{24n})$

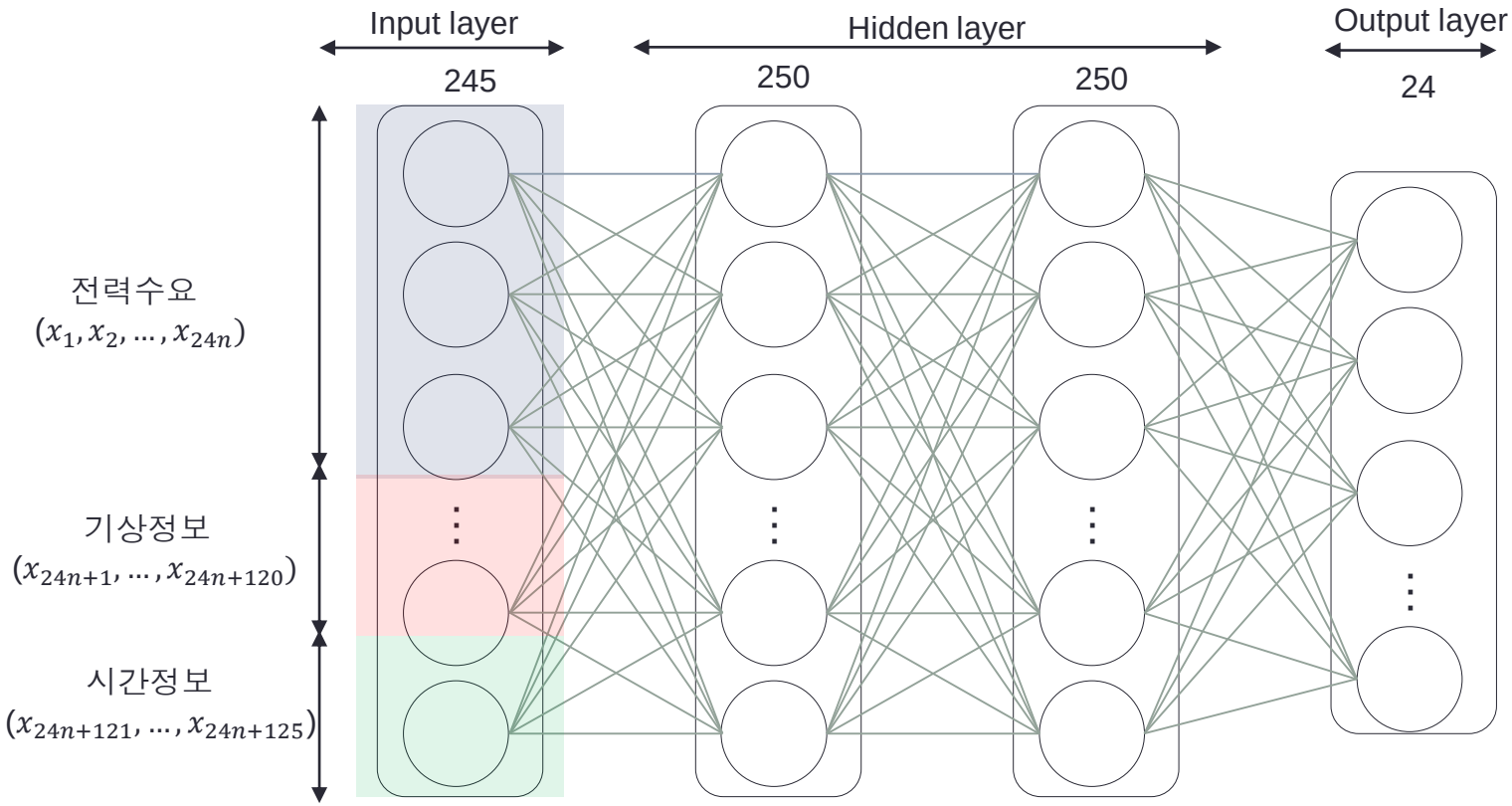
예측전력수요 (label, $y_1, y_2, \dots, y_{24}$)

n=5
trainX : 2034 X 245
trainY : 2034 X 24
testX : 268 X 245
testY : 268 X 24

# DNN 모델

2) 시간별

② Neural Net 구조 (n=5)



# RNN-LSTM 모델

1) 일별

① Input & Output data

| Parameter | Explanation | Number | Dimension |
|---|---|---|---|
| $y$ | 예측일 전력수요 예측 값 | 1 | 1X1 |
| $x_1, x_2, \ldots, x_n$ | 과거 n일동안 일별<br>월, 일, 요일, 휴일여부, 기상정보(5), 전력수요 | n | nX10 |

# RNN-LSTM 모델

1) 일별

① Input & Output data

· data set 구성 코드

```
# build datasets
def build_dataset(time_series, seq_length):
    dataX = []
    dataY = []
    for i in range(0, len(time_series) - seq_length):
        _x = time_series[i:i + seq_length, :]
        _y = time_series[i + seq_length, [-1]]  # Next
        print(_x, "->", _y)
        #print(_y.shape)
        if(_y!=0) :
            dataX.append(_x)
            dataY.append(_y)
    return np.array(dataX), np.array(dataY)
```

$x_1, x_2, \ldots, x_n$

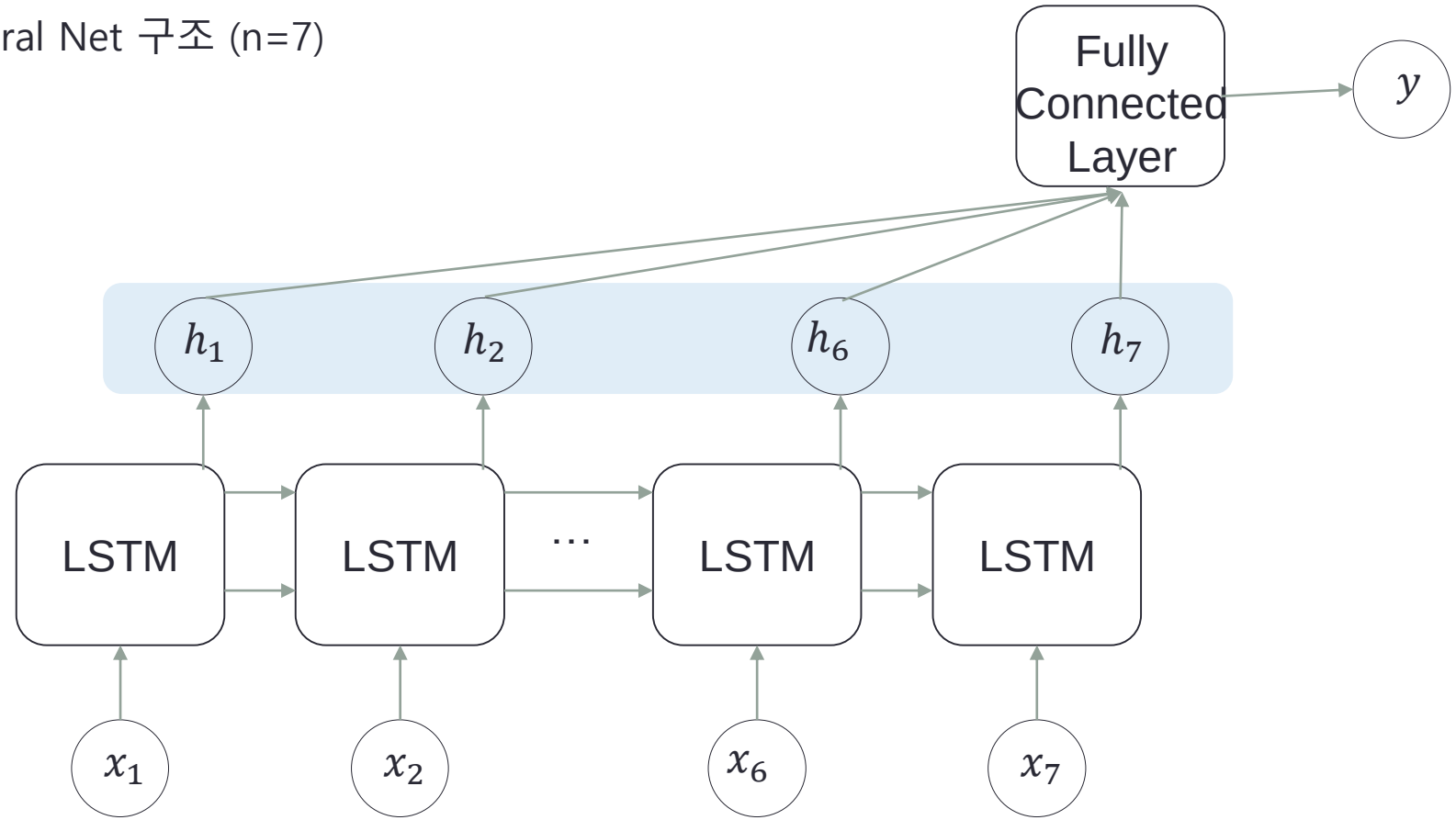
$y$

n=7
trainX : 3644 X 7 X 10
trainY : 3644 X 1
testX : 982 X 7 X 10
testY : 982 X 1

# RNN-LSTM 모델

1) 일별

② Neural Net 구조 (n=7)



# RNN-LSTM 모델

2) 시간별

① Input & Output data

| Parameter | Explanation | Number | Dimension |
|---|---|---|---|
| $y_1, y_2, \dots, y_{24}$ | 24시간 전력수요 예측 값 | 24 | 1X24 |
| $x_1, x_2, \dots, x_{24n}$ | 과거 n일동안 시간별<br>월, 일, 요일, 휴일여부, 기상정보(5), 전력수요 | 24n | 24nX10 |

# RNN-LSTM 모델

2) 시간별

① Input & Output data

· data set 구성 코드

```
# build datasets
def build_dataset(time_series, seq_length):
    dataX = []
    dataY = []

    num = len(time_series)
    i=0
    while i <= num-hseq_length-24:
        for t in range(24):
            if time_series[i+hseq_length+t][11]==0:
                i+=24
                continue

        _x = time_series[i:i+hseq_length, 1:12]
        _y = []
        for j in range(24):
            _y.append(time_series[i+hseq_length+j][-1])

        print(_x, "->", _y)
        dataX.append(_x)
        dataY.append(_y)

        i+=24

    return np.array(dataX), np.array(dataY)
```

$x_1, x_2, \ldots, x_{24n}$

$y_1, y_2, \ldots, y_{24}$

n=5

trainX : 2034 X 120 X 11

trainY : 2034 X 24

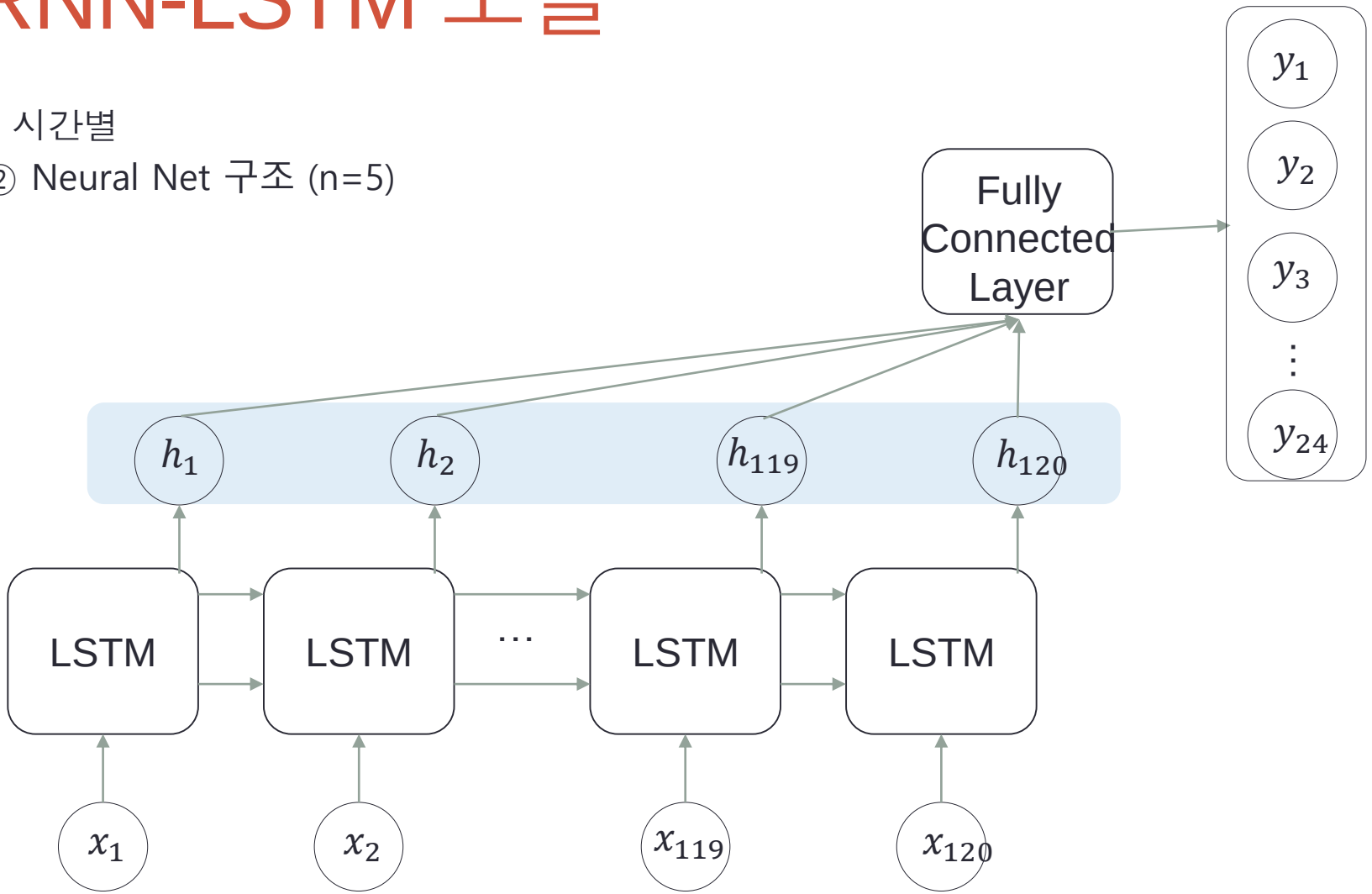
testX : 268 X 120 X 11

testY : 268 X 24

# RNN-LSTM 모델

2) 시간별

② Neural Net 구조 (n=5)

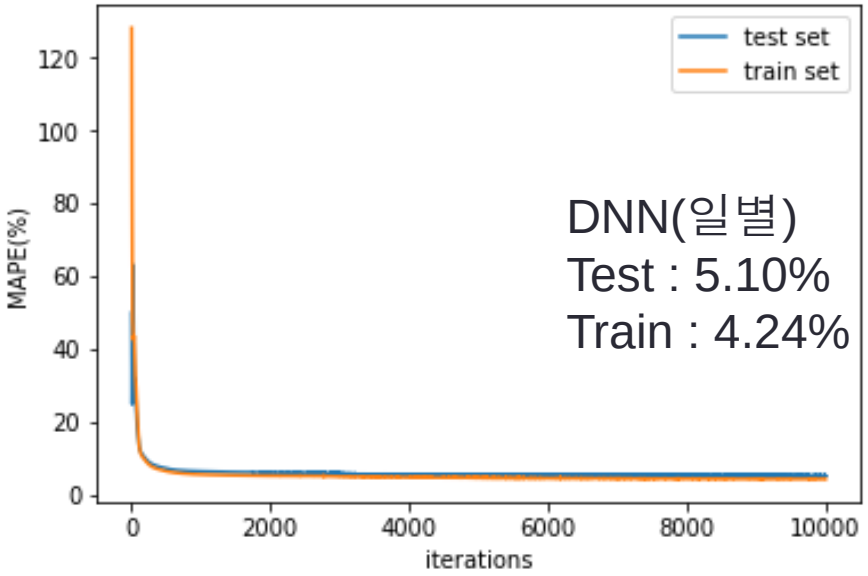


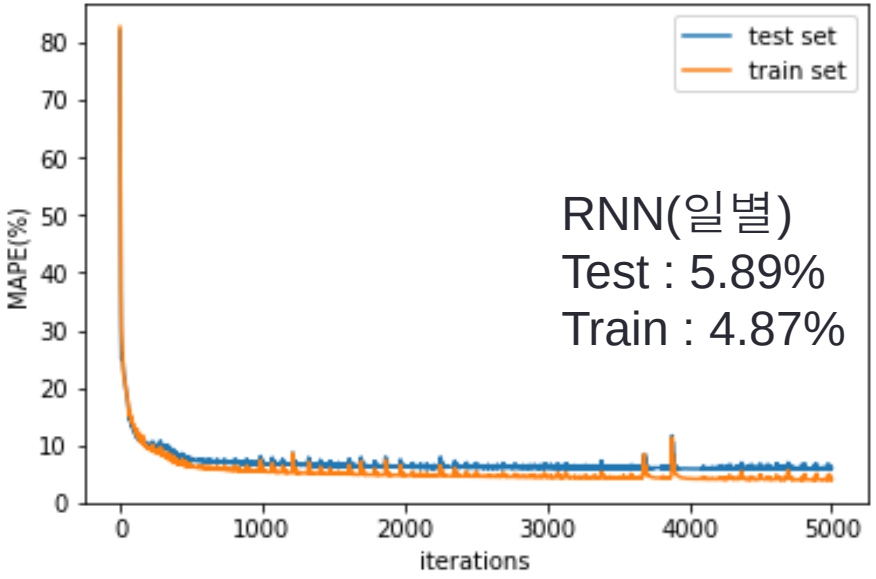
# 성능평가

- MAPE (Mean Absolute Percentage Error)
  예측의 오차율을 측정하기 위해 MAPE 이용

$$mape\ (\%) = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100$$

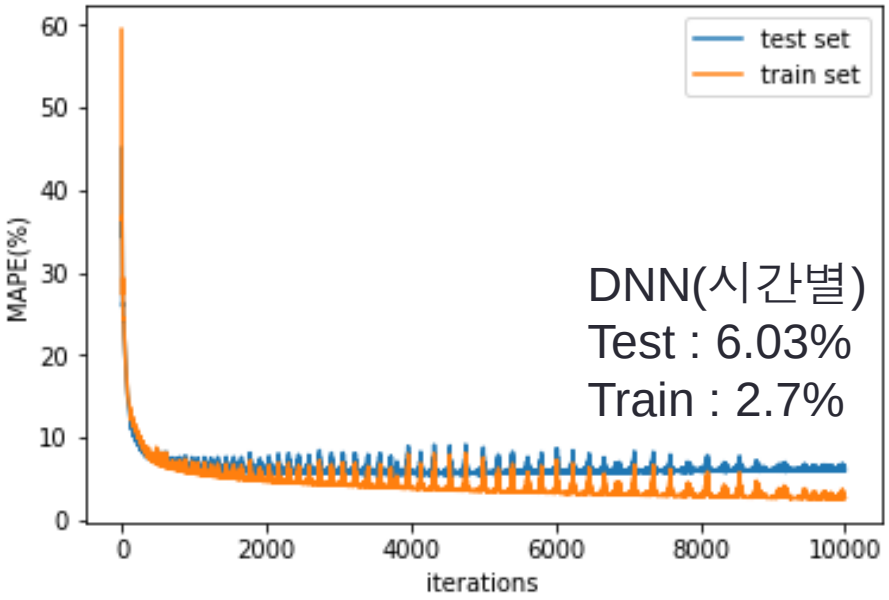
- 일별 DNN, RNN 모델

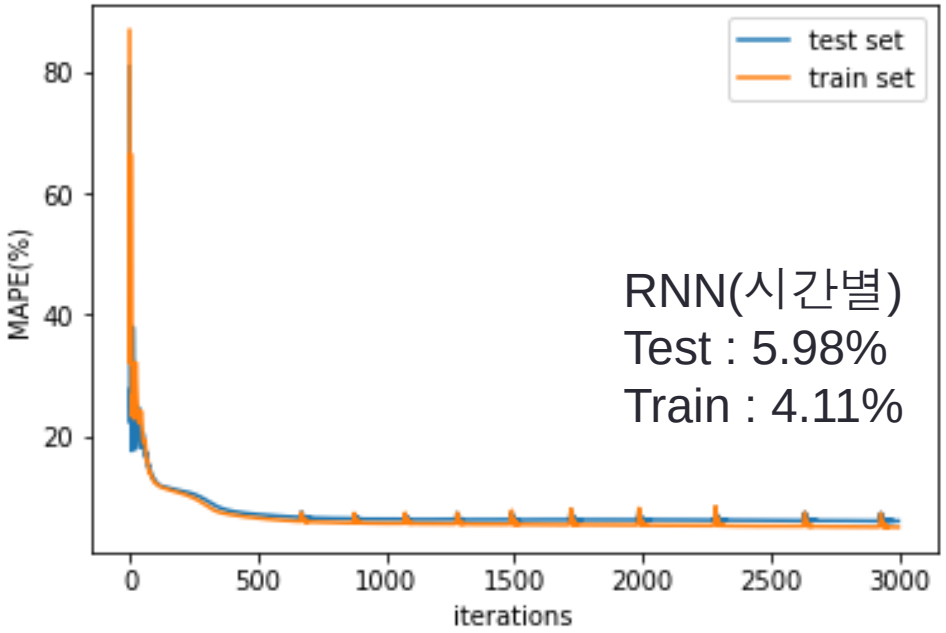


# 성능평가

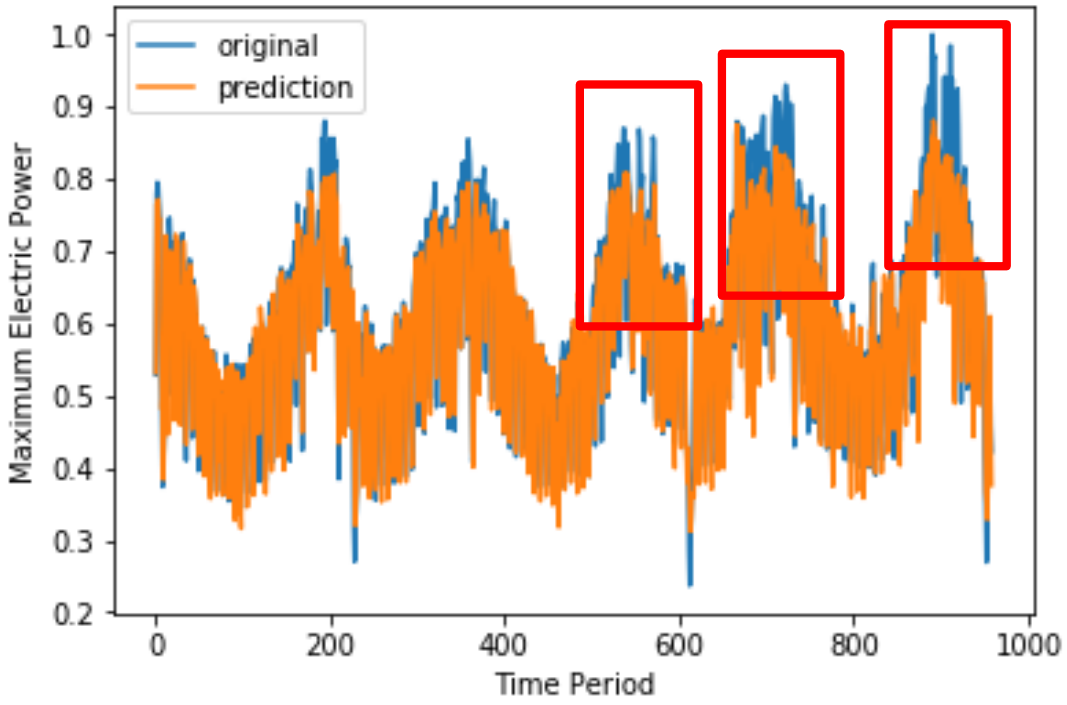
- 시간별 DNN, RNN 모델



# 성능평가

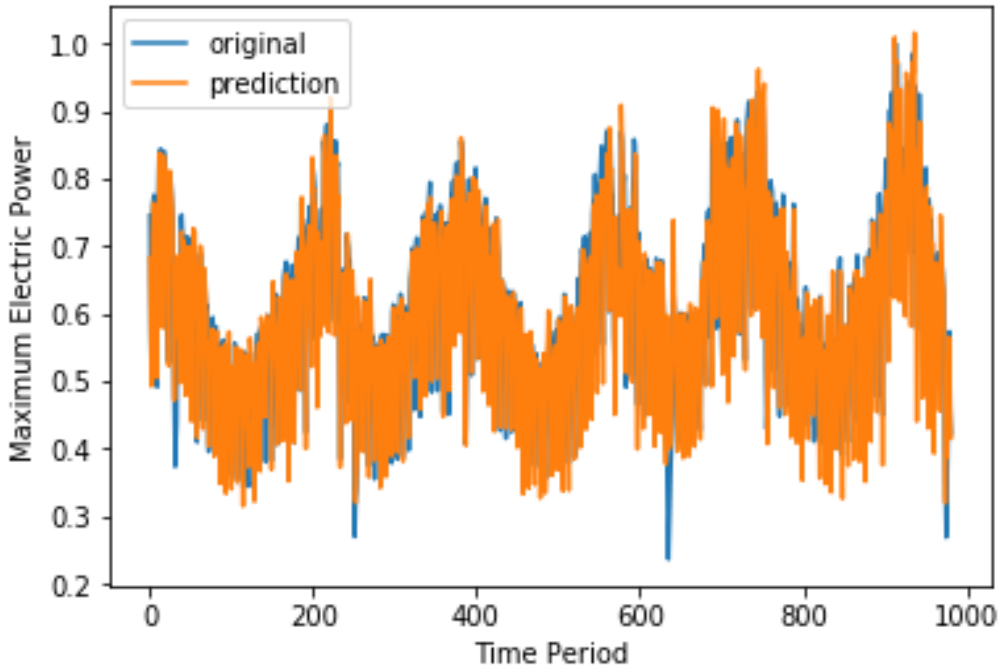
- DNN (일별)




전력수요가 높은 여름과 겨울에
상대적으로 오차율이 크다

# 성능평가

- RNN (일별)




DNN에 비해 균등하게 오차율이 분포하고 있는 것을 알 수 있다.

# 향후 계획

1) PCA, Autoencoder 등을 통한 차원 축소

2) ARIMA 등 통계적 모형과 비교

3) 계절별 성능평가

4) RNN 변형 모델 시도