

LSTM을 이용한 기상 관측 정보 예측 모델 개발

(The Weather Sensory Data Prediction based on LSTM)

지도교수 : 이광근

이 논문을 공학학사 학위 논문으로 제출함.

2020년 6월 26일

서울대학교 공과대학

자유전공학부

유석모

2020년 8월

목차

국문 초록	2
Abstract	2
1. 서론	2
2. 관련 연구	2
3. 방법론	3
3.1. 데이터	3
3.2. 학습 모델	3
3.2.1. 개요	3
3.2.2. 모델 구성	3
3.2.3. 개발 환경 및 소스 코드	5
4. 학습 및 결과 분석	6
4.1. 학습 과정	6
4.2. 분석	8
4.2.1. 평균 기온	8
4.2.2. 최저 기온	10
4.2.3. 최고 기온	12
4.2.4. 일일 강수량	14
5. 결론	16
6. 부록	17
6.1. 관측지점 상세정보	17
참고 문헌	18

국문 초록

시계열 데이터 예측에 효과적인 Long Short-Term Memory(Sepp Hochreiter et al. 1997)을 기반으로 기상 관측 정보 중 기온 및 강수량에 대한 간단한 예측 모델 개발을 주제로 한다. 해당 모델을 통해 관측 데이터를 학습하고 도출된 결과를 수치 분석하여 실제 데이터와의 유사도와 예측 정확도에 대해 논해보고자 한다.

Abstract

Based on Long Short-Term Memory(Sepp Hochreiter et al. 1997), which is effective model to predict time-series data, I developed a simple prediction model of temperature and precipitation in weather sensory data. After training the model by observed data, I compared the result with the real data and analyzed the error of the result by numerical methods to discuss about accuracy.

키워드: 기계학습, 기상예측, 시계열분석, LSTM

1. 서론

기상 관측은 실생활에 매우 중요하지만 예측하기 어려운 정보로 이루어져 있다. 기상 현상은 대규모의 유체의 흐름으로 인해 발생하는 혼돈 시스템이며 물리적으로 해석하기 매우 어려운 현상이다. 수치를 통한 기상 예측 시도는 과거부터 많은 사례가 있어왔다. 회귀분석과 같은 통계적 접근과 직접적인 유체 시뮬레이션까지 다양한 시도가 있었지만 이러한 시도들은 슈퍼컴퓨터에 해당하는 대규모의 컴퓨팅 파워가 요구되어 왔다(박선기 2004, pp.19-20). 최근 들어 클라우드 컴퓨팅의 발달과 기계학습 분야의 비약적인 발전으로 인해 개인에 의한 기상 현상 예측이 불가능한 것만은 아니게 되었다. 1997년 Sepp Hochreiter et al.이 발표한 논문 “LONG SHORT-TERM MEMORY”는 이를 구현하기에 적합한 대표적인 모델이다. 해당 모델은 시계열 데이터의 분석 및 예측에 적합한 구조로 논문 게재 이래 기상 예측(Alireza Koochali et al. 2019) 및 주식 시장 예측(Kang Zhang et al. 2019)에서 자연어 처리(Shuohang Wang et al., 2016)에 이르기까지 다양한 분야에 활용되고 있다. 이 논문에선 Long Short-Term Memory(이하 “LSTM”)을 이용해 기상 관측 정보 중 기온 및 강수량에 대한 간단한 예측 모델을 개발하고자 한다.

2. 관련 연구

Alireza Koochali et al.의 2019년 논문에서 Conditional Generative Adversarial Network(CGAN)의 Generator로 bidirectional LSTM을 사용해 기상 예측 모델을 개발한 사례가 있다. 또한 Kang Zhang et al. 2016과 Shouhang Wang et al. 2016 논문 또한 LSTM을 통한 시계열 예측과 단어 예측을 다루고 있어 LSTM의 실제 활용 사례와 충분한 정확도가 확보될 수 있음을 확인하였고 LSTM과 fully-connected layer를 통한 단순 모델에 대해 정확도가 어느 정도 확보될 지 확인하고자 연구를 진행하였다.

3. 방법론

3.1. 데이터

학습에 사용된 모든 데이터는 기상청 기상자료개방포털¹을 활용하여 작성되었다. 종관기상관측(ASOS) 데이터의 일 단위 데이터를 요청하여 수집하였으며 분석 기간은 1907년 7월 1일부터 2020년 6월 9일까지이다. 해당 기간 중 1950년 1월 1일부터 1953년 12월 31일까지는 한국전쟁으로 인해 데이터 수집이 불완전한 구간으로서 효과적인 모델 개발을 위해 배제하였다. 관측 지점은 서울 108지점으로(위도: 37.57142도, 경도: 126.9658도, 해발고도 86m) 자세한 정보는 6.1.에 기술하였다.

사용된 데이터의 종류는 총 4가지로 평균 기온(섭씨), 최저 기온(섭씨), 최고 기온(섭씨) 및 일강수량(mm)(이하 “목표 수치”)이 사용되었다. 입력은 날짜와 목표 수치에 해당하는 2차원 데이터, 출력은 목표 수치에 해당하는 1차원 데이터로 구성되었다.

3.2. 학습 모델

3.2.1. 개요

LSTM은 Long Short-Term Memory(Sepp Hochreiter et al. 1997)의 약자로 RNN(Recurrent Neural Network)의 일종이다. 내부적으로 지니고 있는 신경망을 통해 이전의 입력을 누적하여 학습한다. 누적된 입력의 이점은 시계열 데이터와 같이 지속적으로 갱신되는 스트림을 처리하기에 유리하다는 점이다. 내부 신경망은 오래된 데이터가 점차적으로 희석되며 최근 데이터의 비중이 높아지는 특징을 지닌다. 해당 모델은 LSTM과 fully-connected layer를 결합한 단순한 구조로 구성되었다.

3.2.2. 모델 구성

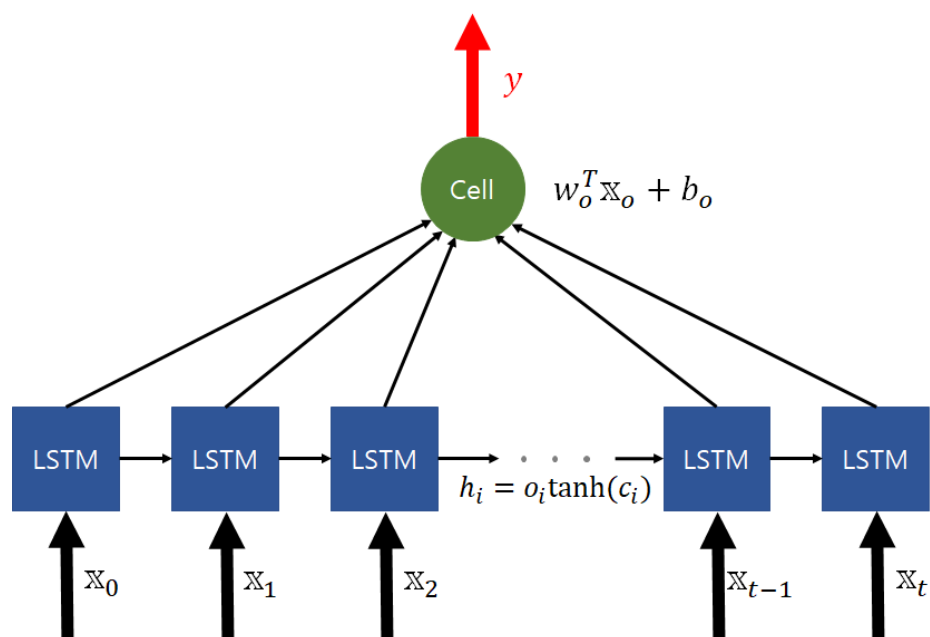


그림 1. LSTM으로 구성된 모델 구조

¹ <https://data.kma.go.kr/cmmn/main.do>, 기상자료개방포털, 2020년 6월 10일 확인.

```
# LSTM model
model = models.Sequential([
    layers.LSTM(units=100, return_sequences=False, dtype=tf.float32, input_shape=(window_size, 1)),
    layers.Dense(units=1, activation=None)
])
model.compile(optimizer='adam', loss='mean_squared_error')
```

그림 2. 모델 구성 Python Code

모델은 100개의 LSTM 유닛으로 구성된다. 각각의 LSTM은 가장 최근부터 100 step 과거의 데이터를 입력으로 가지며 계산 결과는 하나의 Cell로 fully-connected(dense) 되어 최종 출력 $y(w_o^T \mathbf{x}_o + b_o)$ 가 생성된다. 각각의 LSTM은 연결된 이전 LSTM으로부터 은닉 계층의 출력을 받아 다음 LSTM의 weight를 학습한다($h_i = o_i \tanh(c_i)$).

모델에 학습되는 데이터 \mathbf{x}_t 는 30차원(window size)으로 평균 기온(섭씨), 최저 기온(섭씨), 최고 기온(섭씨), 강수량(mm) 4가지 중 하나가 선택되어 sliding window를 이동시키며 학습 데이터를 형성한다. 이는 29차원의 input data(관측 데이터)와 1차원의 target data(예측 출력)로 구성된다.

Date	AvgTemp	Date	AvgTemp	Date	AvgTemp	Date	AvgTemp
1907-10-01	13.5	1907-10-01	13.5	1907-10-01	13.5	1907-10-01	13.5
1907-10-02	16.2	1907-10-02	16.2	1907-10-02	16.2	1907-10-02	16.2
1907-10-03	16.2	1907-10-03	16.2	1907-10-03	16.2	1907-10-03	16.2
1907-10-04	16.5	1907-10-04	16.5	1907-10-04	16.5	1907-10-04	16.5
1907-10-05	17.6	1907-10-05	17.6	1907-10-05	17.6	1907-10-05	17.6
1907-10-06	13	1907-10-06	13	1907-10-06	13	1907-10-06	13
1907-10-07	11.3	1907-10-07	11.3	1907-10-07	11.3	1907-10-07	11.3
1907-10-08	8.9	1907-10-08	8.9	1907-10-08	8.9	1907-10-08	8.9
1907-10-09	11.6	1907-10-09	11.6	1907-10-09	11.6	1907-10-09	11.6
1907-10-10	14.2	1907-10-10	14.2	1907-10-10	14.2	1907-10-10	14.2
1907-10-11	15.4	1907-10-11	15.4	1907-10-11	15.4	1907-10-11	15.4
1907-10-12	13.9	1907-10-12	13.9	1907-10-12	13.9	1907-10-12	13.9
1907-10-13	13.8	1907-10-13	13.8	1907-10-13	13.8	1907-10-13	13.8
1907-10-14	13	1907-10-14	13	1907-10-14	13	1907-10-14	13
1907-10-15	13.1	1907-10-15	13.1	1907-10-15	13.1	1907-10-15	13.1
1907-10-16	14.1	1907-10-16	14.1	1907-10-16	14.1	1907-10-16	14.1
1907-10-17	16.4	1907-10-17	16.4	1907-10-17	16.4	1907-10-17	16.4

그림 3. Sliding window를 통한 학습 데이터 형성 예시(window size=10일 때)

```

# split data by sliding window
def sliding_window_split(data, window_size):
    X = np.zeros((len(data) - window_size, window_size, 1))
    y = np.zeros((len(data) - window_size, 1))
    for i in range(0, len(data) - window_size):
        partial_x = data[i:i + window_size].reshape((-1, 1)) # window_size-1 of data
        partial_y = data[i + window_size].reshape((-1)) # 1 of data
        X[i], y[i] = partial_x, partial_y
    return X, y

```

그림 4. Sliding window 학습 데이터 형성 Python Code

3.2.3. 개발 환경 및 소스 코드

Python 3.6.9 언어로 TensorFlow 2.2.0 library을 이용하여 18.04.3 LTS OS의 Google Colab 환경에서 개발되었다. 실제 개발 코드는 아래 주소에서 확인할 수 있다.

<https://github.com/trouvaille/snu2020-graduation-thesis>

4. 학습 및 결과 분석

4.1. 학습 과정

기상자료개방포털의 서울 108지점 ASOS 데이터 중 1907년 7월 1일부터 2020년 6월 9일(한국전쟁으로 인한 1950년 1월 1일부터 1953년 12월 31일 데이터 소거)까지의 39700개의 데이터를 입력으로 사용하였다. LSTM의 input으로 들어가는 Sliding window에 사용된 window_size는 한 달의 주기성과 관련하여 30으로 설정하였다. 39700개의 데이터 중 50%의 데이터(19850개)는 학습에 사용되었고, 나머지 50% 데이터(19850개)는 테스트에 사용되었다. Loss의 감소가 현저히 줄어드는 Epoch 수의 경계를 관측하여 Epoch 수는 10으로 설정하였다.

```
# set up data path and training constants
#data_path = "data/OBS_ASOS_DD_19071001-20200609.csv"
data_path = "/content/snu2020-thesis/data/OBS_ASOS_DD_19071001-20200609.csv"
train_ratio = 0.5 # ratio of training
window_size = 30 # sliding window size(day)
num_epochs = 10 # number of epoches
```

그림 5. 학습 매개변수 설정

평균 기온(섭씨), 최저 기온(섭씨), 최고 기온(섭씨), 강수량(mm) 4가지 열 중 하나만을 선택해 학습을 진행한다(1차원 데이터). 먼저 median filter를 사용해 데이터의 편차를 줄여서 학습의 정확도를 상승시키고자 하였다. 그 후 각 데이터의 최저 값과 최고 값을 균등하게 $[-1, 1]$ 범위로 scaling하여 normalization을 시행하였다. 데이터가 TensorFlow model에 입력될 때 정규화 된 데이터가 더 나은 학습결과를 가져온다.

```
# read data and preparation
data_raw = pd.read_csv(data_path) # read csv raw data
data_raw = data_raw.fillna(0) # fill NaN 0
data_raw = data_raw.iloc[:,1:2] # use only AvgTemp
#data_raw = data_raw.iloc[:,2:3] # use only LowTemp
#data_raw = data_raw.iloc[:,3:4] # use only HighTemp
#data_raw = data_raw.iloc[:,4:5] # use only Precipitation
data_raw = data_raw.to_numpy() # convert to numpy
data = medfilt(data_raw.reshape((-1)), 3).reshape((-1,1)) # smoothing by median filter
data, mins, scales = normalize(data) # normalize
```

그림 6. Data preparation

```

# normalize data
def normalize(data):
    res = np.zeros(data.shape)
    row, col = data.shape
    maxs = np.copy(data[0])
    mins = np.copy(data[0])
    scales = np.zeros(col)
    # find min and max
    for i in range(0, row):
        for j in range(0, col):
            if (maxs[j] < data[i, j]): maxs[j] = data[i, j]
            if (mins[j] > data[i, j]): mins[j] = data[i, j]
    # calculate scales
    for j in range(0, col): scales[j] = maxs[j] - mins[j]
    # normalize
    for i in range(0, row):
        for j in range(0, col):
            res[i, j] = (data[i, j] - mins[j]) / scales[j]
            res[i, j] = 2.0 * res[i, j] - 1.0 # range -1 ~ 1
    return res, mins, scales

```

그림 7. 정규화 함수

정규화된 데이터는 3.2.2.의 train_ratio의 비율에 따라 train_set과 test_set으로 나누어진 뒤에 sliding window를 통해 input과 target 데이터로 분리된다.

```

# split training and test data
train_set = data[0:int(data.shape[0] * (1-train_ratio)),:]
train_set_raw = data_raw[0:int(data.shape[0] * (1-train_ratio)),:]
test_set = data[int(data.shape[0] * (1-train_ratio)):data.shape[0],:]
test_set_raw = data_raw[int(data.shape[0] * (1-train_ratio)):data.shape[0],:]

# split data into window size
X_train, y_train = sliding_window_split(train_set, window_size)
X_train_raw, y_train_raw = sliding_window_split(train_set_raw, window_size)
X_test, y_test = sliding_window_split(test_set, window_size)
X_test_raw, y_test_raw = sliding_window_split(test_set_raw, window_size)

```

그림 8. 학습 및 테스트 데이터 분리 및 sliding window input 및 target 데이터 형성

최종적으로 가공된 X_train과 y_train을 통해 128 크기의 batch size 아래에서 모델을 학습시킨다.

```

# train the model
result = model.fit(X_train, y_train, epochs=num_epochs, batch_size=128)
loss = result.history['loss']

```

그림 9. 모델 학습

4.2. 분석

4.2.1. 평균 기온

테스트 세트를 이용해 예측 값을 생성한 뒤 실제 관측 데이터 값과의 오차를 Mean Squared Error(MSE)와 R squared metric을 통해 오차를 측정하였다.

평균 기온은 MSE 3.1986, R_squared 0.9706로 97.1%에 해당하는 정확도를 보였다.

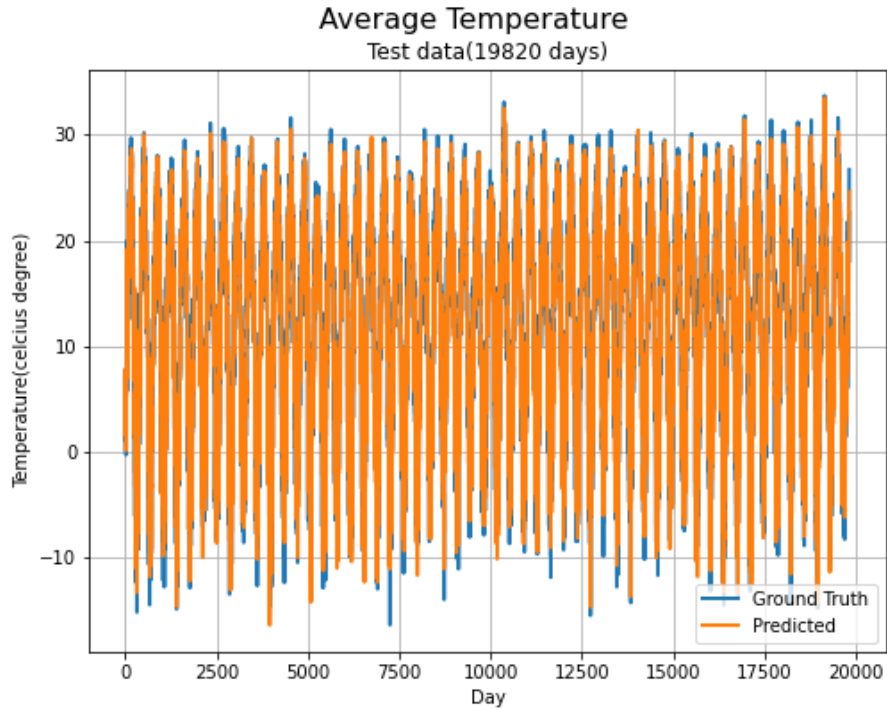


그림 10. 평균 기온 관측 및 예측 그래프(전체 테스트 세트)

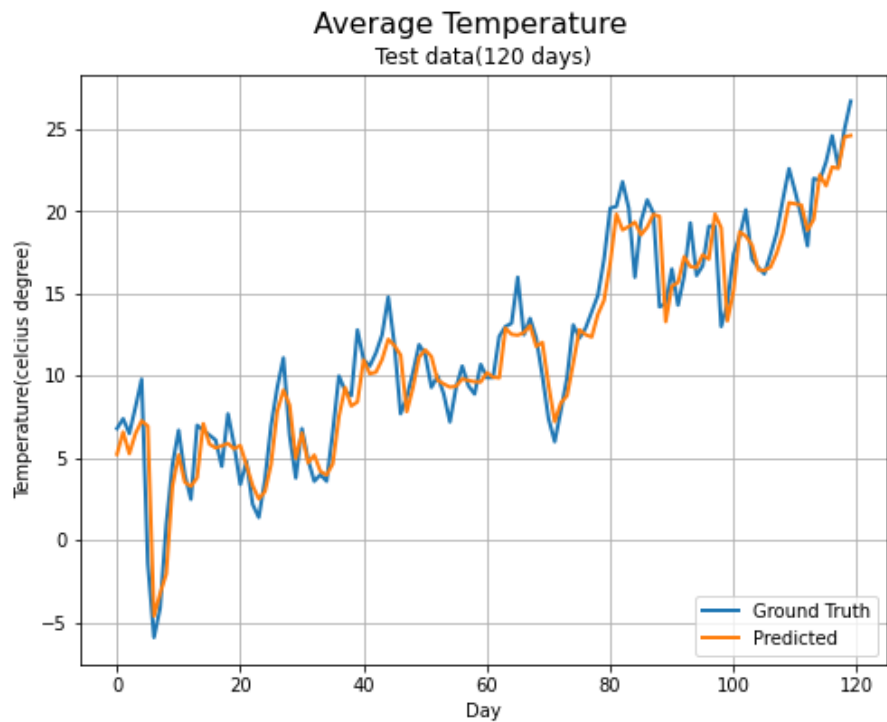


그림 11. 평균 기온 관측 및 예측 그래프(최근 120일)

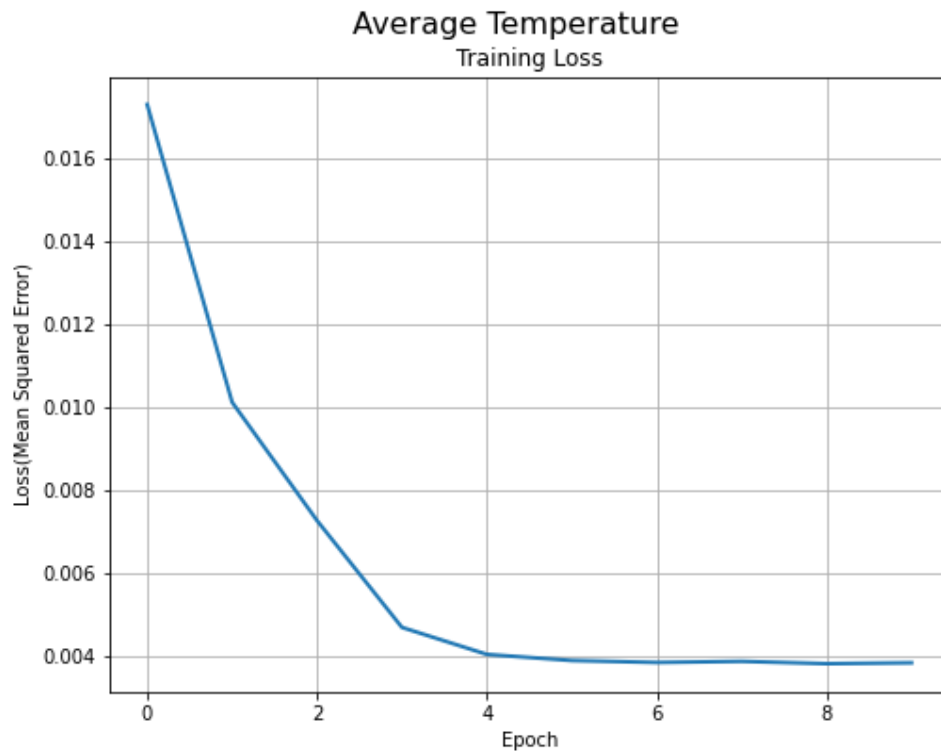


그림 12. 평균 기온 Epoch에 따른 Loss 감소 그래프

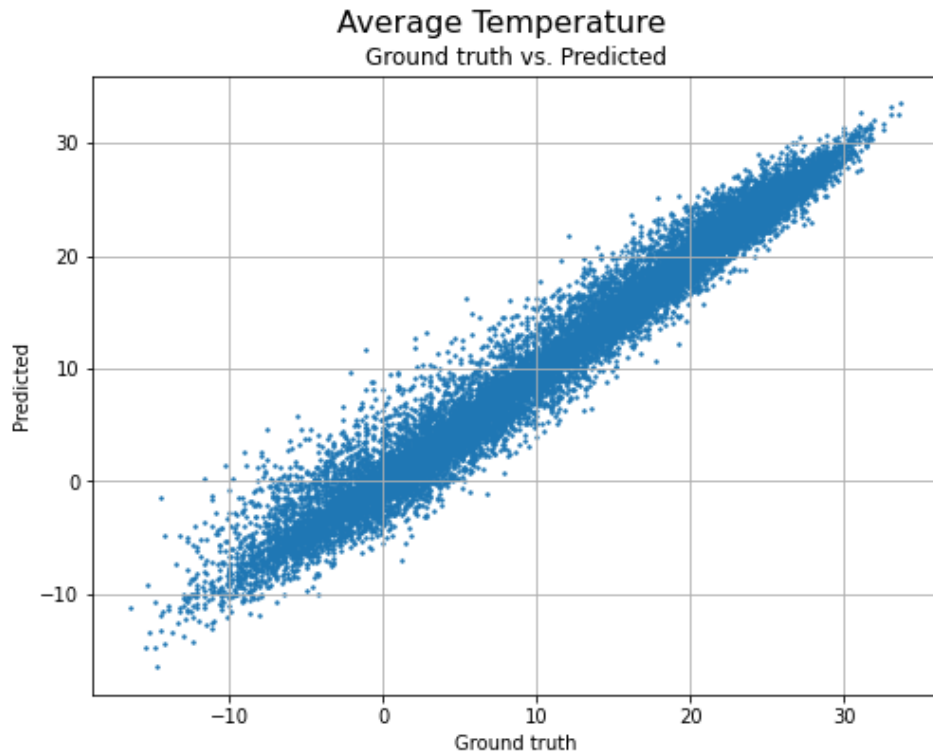


그림 13. 평균 기온 관측 및 예측 산점도

4.2.2. 최저 기온

최저 기온은 MSE 3.7829, R_squared 0.9656로 96.6%에 해당하는 정확도를 보였다.

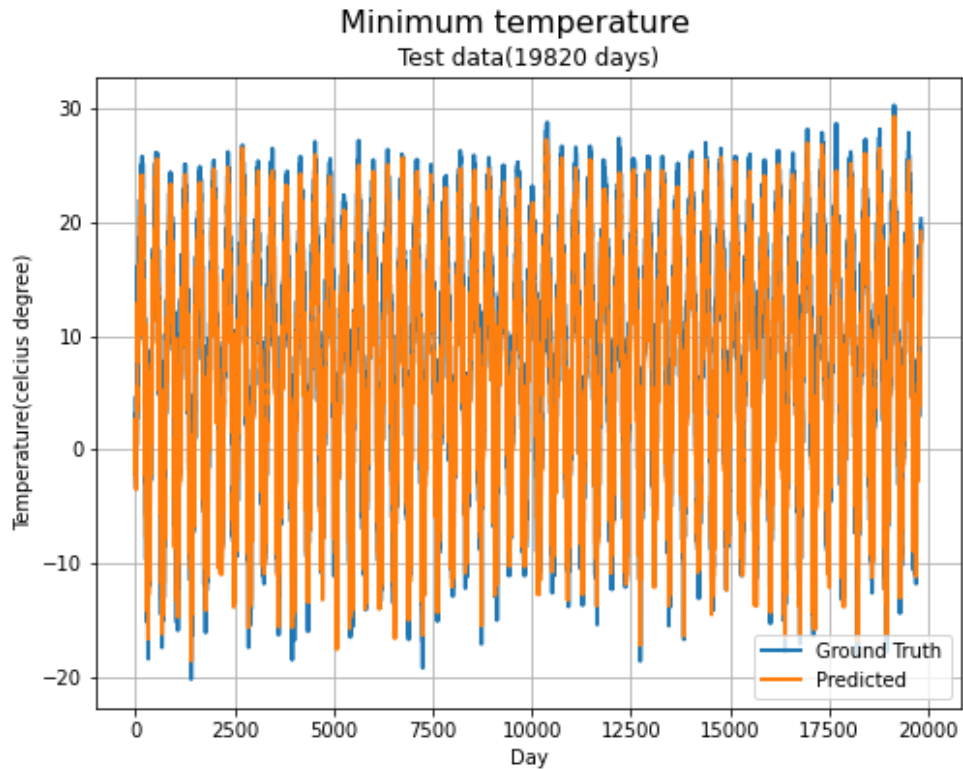


그림 14. 최저 기온 관측 및 예측 그래프(전체 테스트 세트)

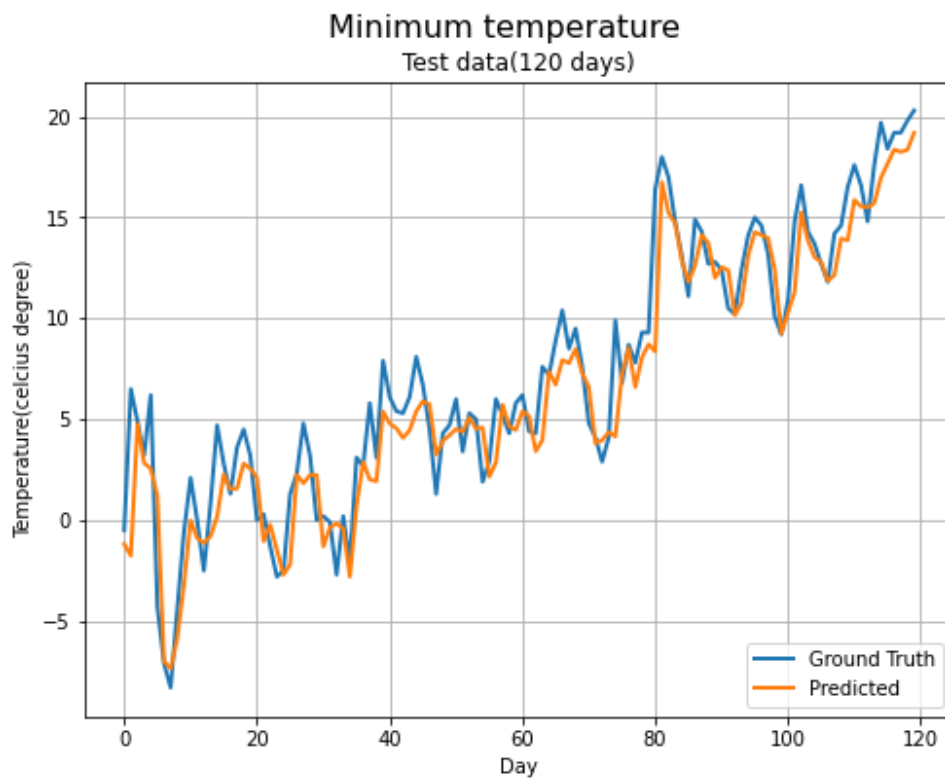


그림 15. 최저 기온 관측 및 예측 그래프(최근 120일)

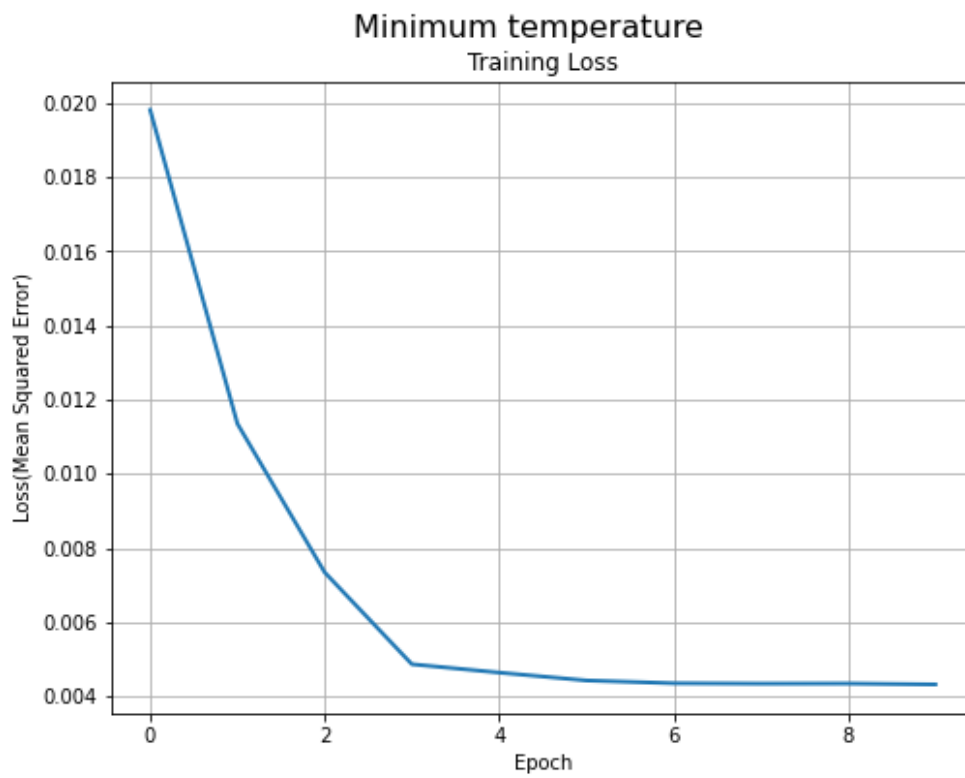


그림 16. 최저 기온 Epoch에 따른 Loss 감소 그래프

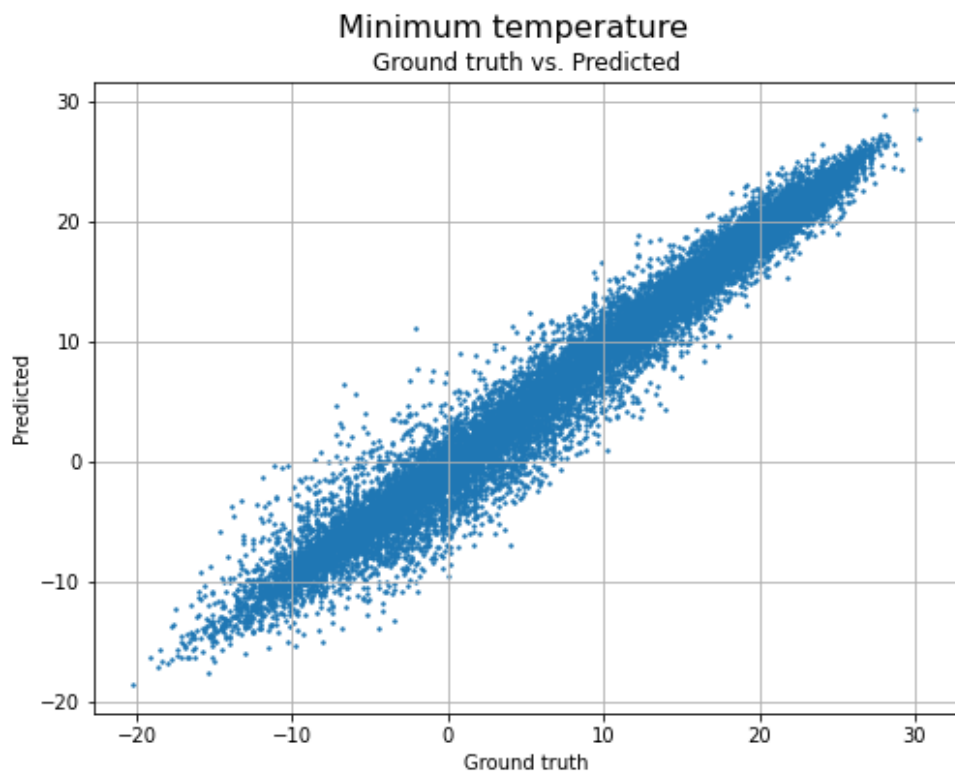


그림 17. 최저 기온 관측 및 예측 산점도

4.2.3. 최고 기온

최고 기온은 MSE 5.3571, R_squared 0.9530로 95.3%에 해당하는 정확도를 보였다.

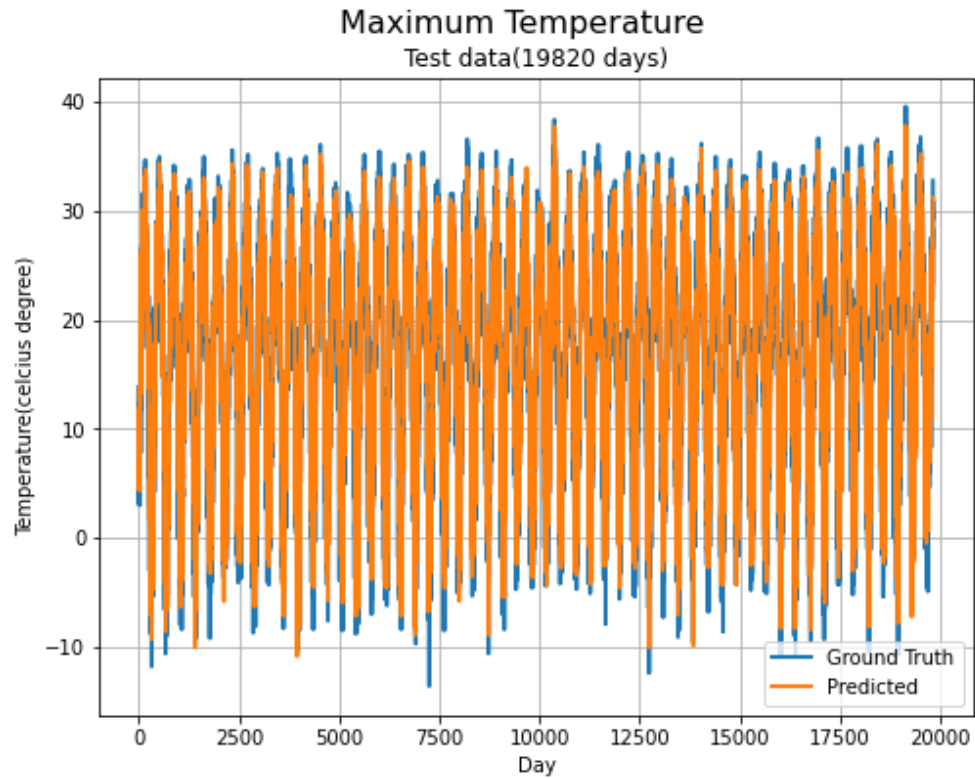


그림 18. 최고 기온 관측 및 예측 그래프(전체 테스트 세트)

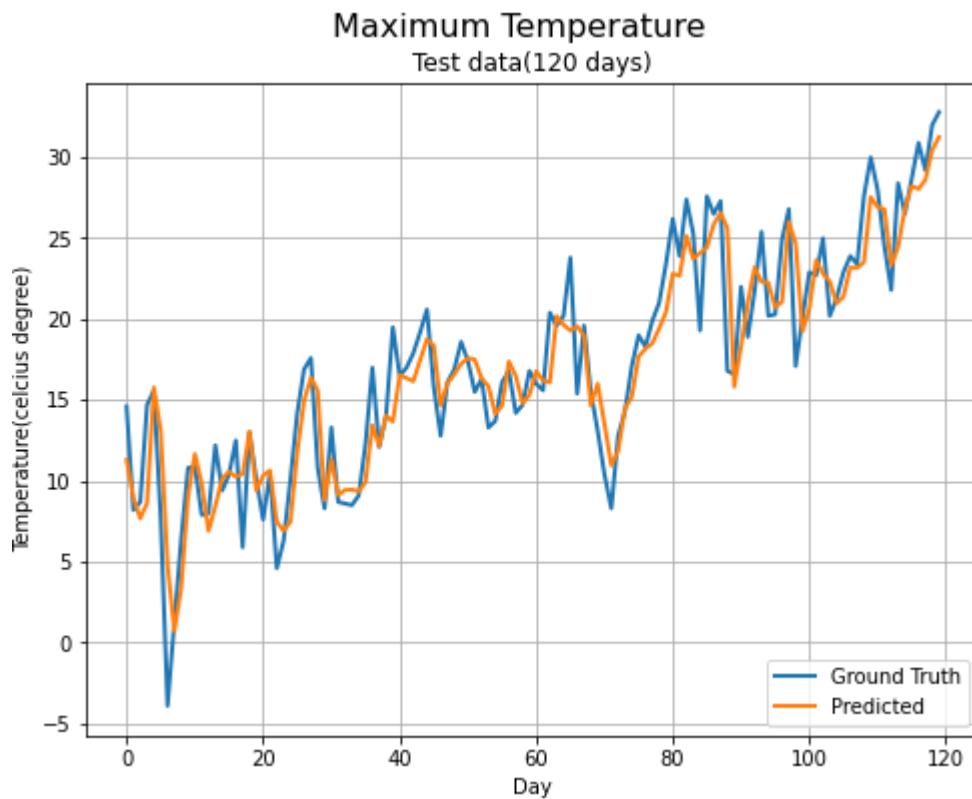


그림 19. 최고 기온 관측 및 예측 그래프(최근 120일)

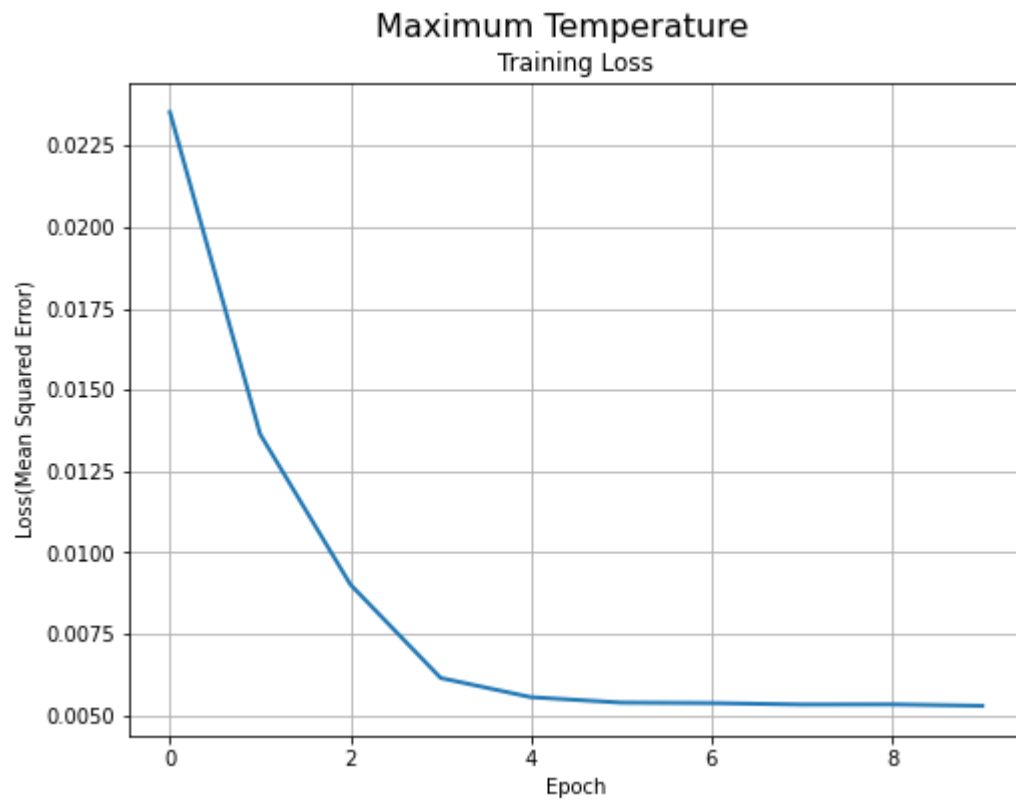


그림 20. 최고 기온 Epoch에 따른 Loss 감소 그래프

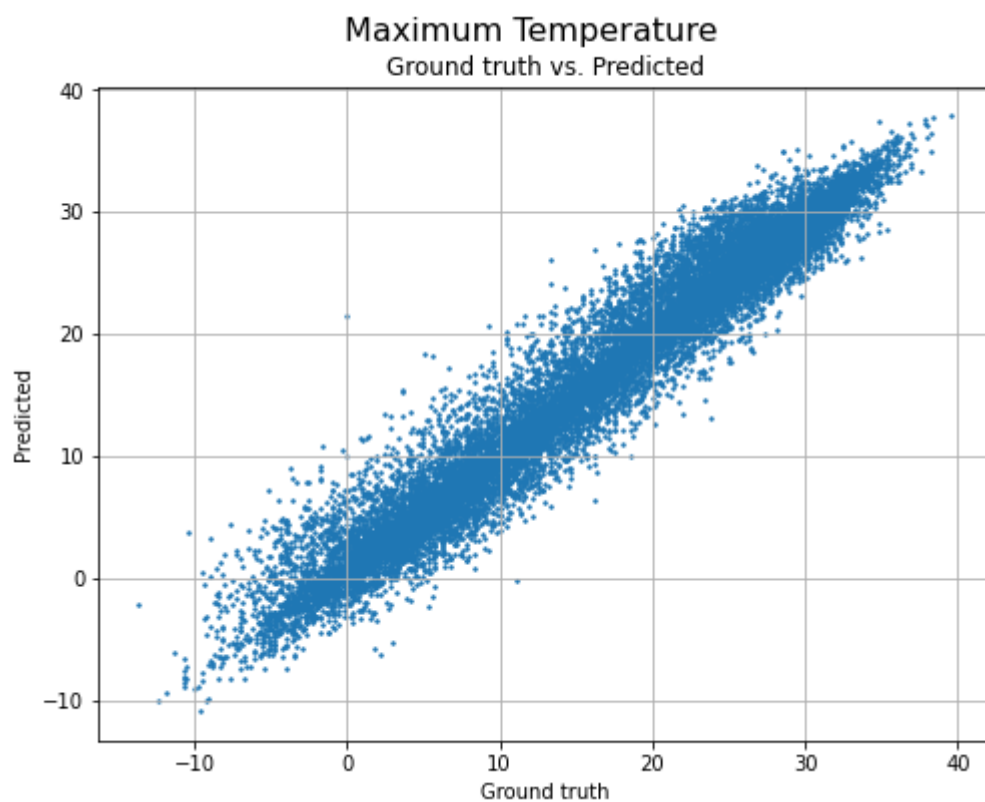


그림 21. 최고 기온 관측 및 예측 산점도

4.2.4. 일일 강수량

MSE 181.0255, R_squared 0.1642로 16.4%에 해당하는 정확도를 보였다.

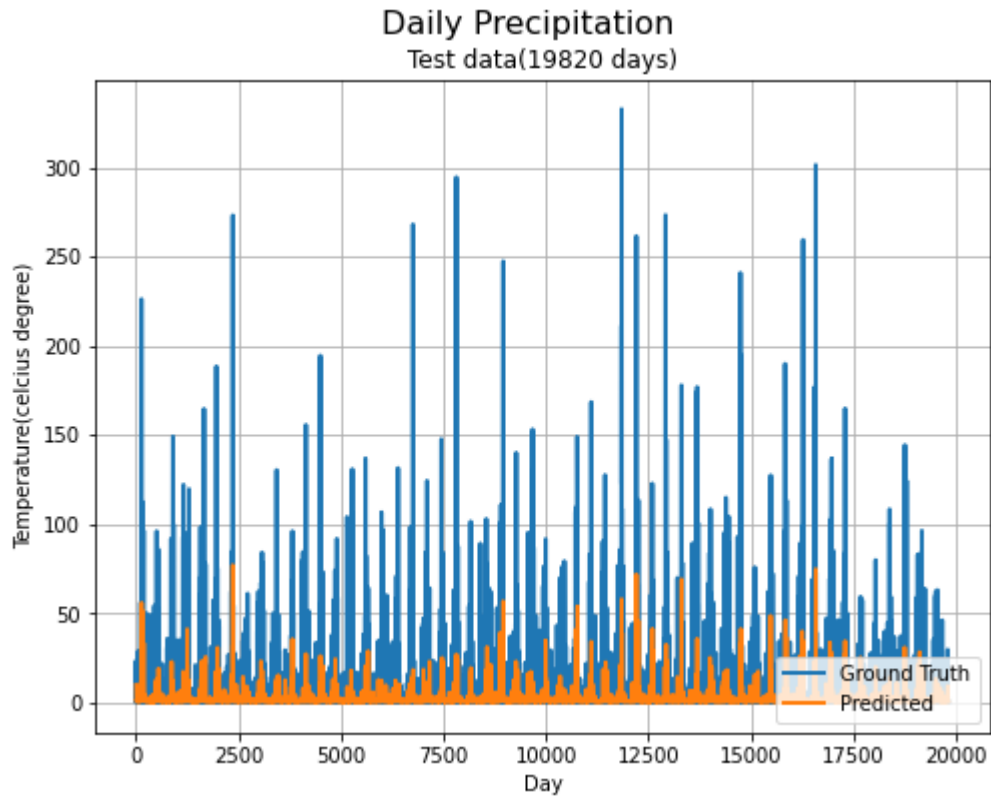


그림 22. 일일 강수량 관측 및 예측 그래프(전체 테스트 세트)

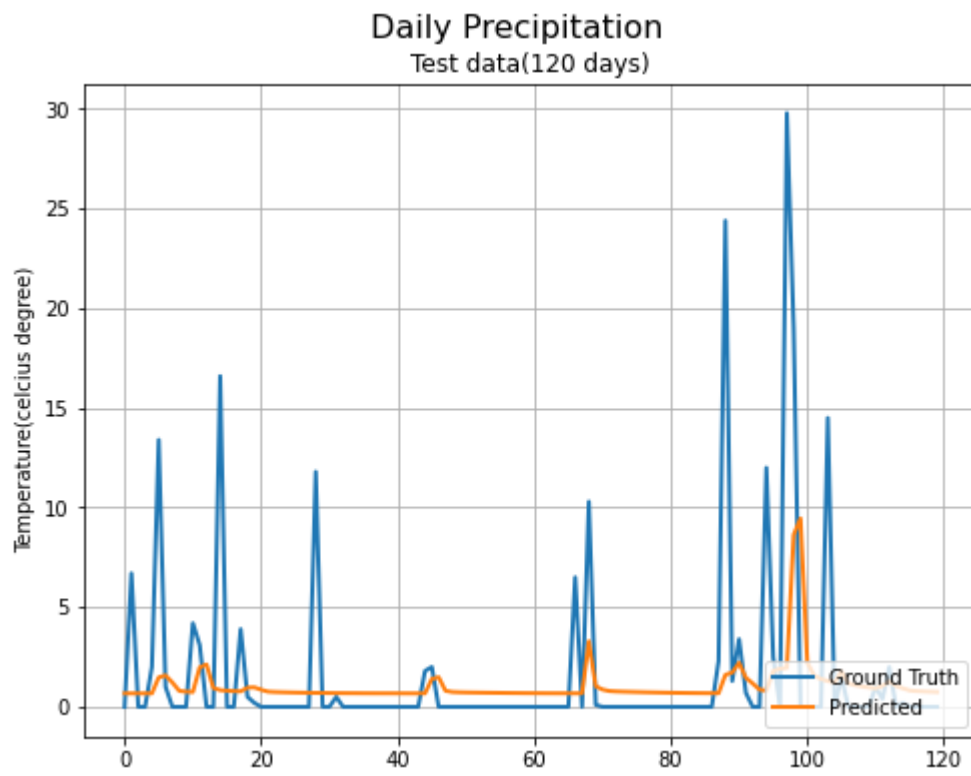


그림 23. 일일 강수량 관측 및 예측 그래프(최근 120일)

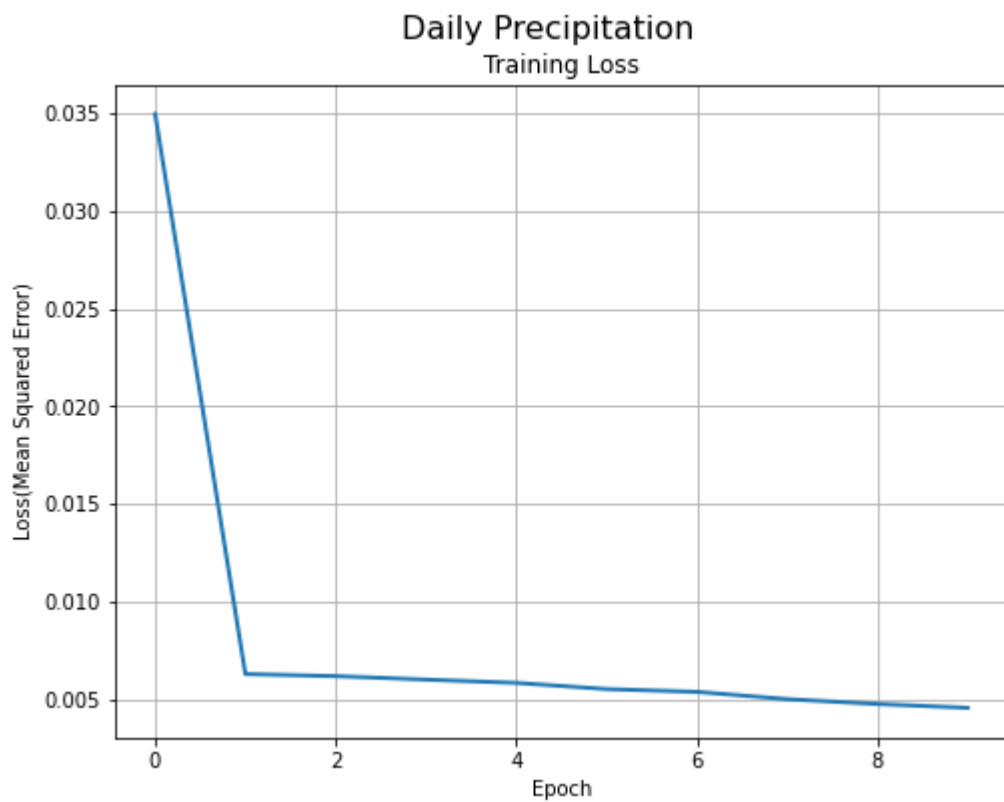


그림 24. 일일 강수량 Epoch에 따른 Loss 감소 그래프

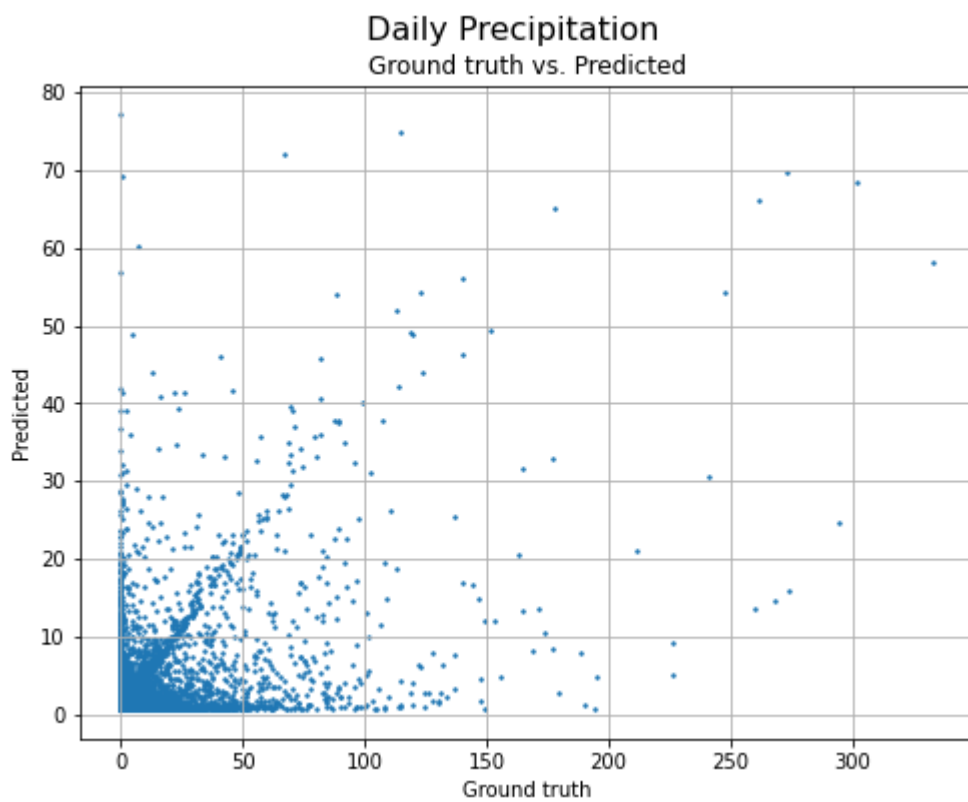


그림 25. 일일 강수량 관측 및 예측 산점도

5. 결론

평균 기온, 최저 기온, 최고 기온과 같은 기온 변수는 LSTM 및 single fully-connected layer로 충분히 효과적인 결과를 얻을 수 있었다(R_squared 0.95 이상). 이는 주목할 만한 결과로 Deep Convolutional Network(DCN)나 Generative Adversarial Network(GAN)와 같은 모델과 결합하여 모델을 구성하면 높은 비율로 신뢰할 만한 데이터를 얻을 수 있으리라 기대되는 점이었다.

하지만 일일 강수량은 LSTM 및 single fully-connected layer의 구성만으론 효과적인 결과를 구하지 못하였다. R_squared 값이 16.4%로 사실상 예측에 실패한 모델이었다. 그림 24를 보면 Epoch 1까지 효과적인 학습이 이루어졌으나 Epoch 2 이후 효과적인 Loss의 감소를 보이지 못해 추가적인 모델 구성이 필요해 보인다. 앞서 언급한 DCN이나 GAN과 같은 모델과 결합하면 보다 나은 결과를 보이는 지 추가적인 연구가 필요해 보인다. 그림 25를 보면 $y=x$ 의 직선과 맞닿아 있는 데이터의 분포가 일정 부분 나타나고 있고 $y=x$ 의 직선과 먼, 예측에 실패한 데이터 또한 넓게 분포하고 있다. 해당 산점도를 통해 LSTM의 출력이 완전히 의미없는 데이터를 의미한다기보다 파라미터의 수정이나 추가적인 레이어 또는 모델의 구성을 통하면 보다 나은 모델 개발이 가능함을 알 수 있다.

LSTM은 이와 같이 오래전 발표된 논문(Sepp Hochreiter et al., 1997)임에도 불구하고, 적합한 데이터 가공과 함께 모델을 구성하면 주기성이 강한 데이터의 경우 상당히 높은 정확도로 데이터를 예측할 수 있음을 확인하였다. 하지만 DCN이나 GAN 같은 보다 효과적인 모델이 많이 개발된 시점에서 LSTM 모델의 단일 사용보다는 다른 모델과 결합하여 사용하는 것이 더욱 높은 정확도를 보이는 시계열 예측 모델을 개발하는데 도움이 될 것이다.

6. 부록

6.1. 관측지점 상세정보²


지점 정보

지점유형	지상	장비명	
관리기관	기상청	운영기관	기상청 수도권기상청 관측과
표준지점번호	108	기관지점번호	108
지점명(한글)	서울	지점명(영문)	
관측개시일	1907-10-01	관측주기(분)	1
좌표(WGS84)	위도 : 37.57142 경도 : 126.9658		
GPS 측정좌표계		GPS 측정지점	표석에서 측정
GPS 측정일시	2005-06-24	해발고도(m)	86
설치목적	기상기후	운영방법	

관측장소 전경

이미지		제목	서울
		촬영일자	2017-05-11
이미지		제목	서울
		촬영날씨	맑음
		촬영일자	2017-05-11

전체 배치도

이미지		제목	서울
		촬영일자	2005-12-02

² <https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36>, 기상자료개방포털[데이터:기상관측:지상:종관기상관측(ASOS)], 2020년 6월 10일 확인.

참고 문헌

- [1] Sepp Hochreiter et al., “LONG SHORT-TERM MEMORY,” 1997.
- [2] 박선기, “기상 및 기후의 수치예측에 대한 슈퍼컴퓨터의 역할,” 2004.
- [3] Alireza Koochali et al., “Probabilistic Forecasting of Sensory Data with Generative Adversarial Networks – ForGAN,” 2019.
- [4] Kang Zhang et al., “Stock Market Prediction Based on Generative Adversarial Network,” 2019.
- [5] Shuohang Wang et al., “Learning Natural Language Inference with LSTM,” 2016.