연구 보고서

1. 연구 소개

1-1. 논문 소개

논문(사회적 변수를 고려한 LSTM 기반 코로나19 일별 확진자 수 예측 기법)의 실험을 재현하고자 한다.

논문은 일별 코로나19 추가 확진자 수를 예측하기 위해 RNN이나 LSTM과 같은 여러 기계학습 모델을 사용하였다. 그리고 시계열 예측 모델을 구성하는 데에 있어서 여러 사회적 변수를 사용하여 성능을 개선되었음을 입증하였다.

논문에서 실험에 사용한 변수는 아래와 같다.

▼ 변수 목록

1. 사회적 거리두기 단계
2. “코로나19” 네이버 검색량
3. 지하철 승객 수
4. 백신 1차 신규 접종량
5. 백신 1차 총 접종량
6. 백신 1차 접종 비율
7. 백신 2차 신규 접종량
8. 백신 2차 총 접종량
9. 백신 2차 접종 비율
10. 평균 기온
11. 평균 지면 온도
12. 30cm 지중 온도
13. 5m 지중 온도
14. 일 미세먼지 농도
15. 합계 일사량
16. Month_x = sin((360/12) × Month)
17. Month_y = cos((360/12) × Month)
18. Day_x = sin((360/cycle) × Day)
19. Day_y = cos((360/cycle) × Day)
20. 서울시 일별 확진자 수

16~19번 변수는 저자의 다른 논문인 논문(LSTM 기반의 코로나 일별 확진자 수 예측 기법)에 따라 날짜의 주기성을 포함하였다.

논문의 실험은 서울시의 일별 코로나19 추가 확진자 수를 예측하는 것이 목표이므로 2번의 “코로나19” 네이버 검색량을 제외한 나머지 변수는 서울시의 데이터이다.

총 20개의 변수를 평균 잠복기인 5일간의 데이터로 묶어 각각 1, 7, 14일 후의 확진자 수를 예측하였다. 이렇게 구성된 한 단위의 데이터는 (5, 20)의 입력과 1개의 정답 label이 존재한다. 이렇게 (5, 20) 크기의 입력 데이터는 sliding window가 하루만큼만 움직이며 만든다고 볼 수 있다.
예를 들어 1~5일의 20개의 변수 데이터를 입력으로 사용하고 6, 12, 19일의 추가 확진자 수를 정답으로 사용한 것이다.

데이터의 기간은 2020년 1월 24일에서 2021년 8월 3일까지(총 558일)이었으며, 이를 8:2의 비율로 train과 test부분으로 나누었다.

LSTM을 구성하기 위해 사용된 정보는 아래의 표와 같다.

손실 함수	MSE(Mean Squared Error)
최적화 함수	Adam(Adaptive Moment Estimation)
셀 크기	16
배치 크기	16
학습률	0.001
학습 반복 횟수	30

각각의 예측 성능을 비교하기 위한 평가 지표로는 RMSE(Root Mean Squared Error)와 PCC(Pearson Correlation Coefficient)를 사용하였다.

1-2. 실험 내용

이 연구의 궁극적인 목표는 논문보다 더 나은 성능의 모델을 만드는 것이다. 즉, 논문보다 코로나19 추가 확진자 수를 예측을 더욱 잘하는 모델을 만드는 것을 의미한다.

이를 위해서는 데이터의 전체 기간을 늘리거나 추가적인 변수 사용으로 학습 데이터의 수를 늘리는 방법이 있고 학습 횟수를 늘리는 방법을 시도해 볼 수 있다.

따라서 이 실험의 목표는 여러 방식으로 논문의 실험을 재현하는 것이다. 이 실험을 위해 논문과 같은 변수를 사용하며 논문 발표 이후의 최신 데이터를 확보할 수 있었다.

그리고 이 실험에서 사용된 인공신경망은 시계열 데이터에 적합한 RNN과 LSTM이다.

1-3. RNN과 LSTM

1-3-1. RNN

RNN(Recurrent Neural Network)은 인공 신경망 중의 하나로, 이름 그대로 순환 신경망이다. 신경망의 출력이 다시 입력으로 재귀적인 연결 구조를 가지고 있다. 입력과 출력이 서로 독립적이지 않다. 그러므로 과거의 정보를 가지고 갈 수 있다. 따라서 데이터의 입력 순서가 중요하다. RNN은 네트워크의 각 계층 내에서 동일한 가중치 매개변수를 공유한다.
RNN의 장점으로는 이전 단계의 정보를 가지고 있어 연속적인 데이터를 처리하기 좋다. 특히 자연어처리, 시계열 예측에 사용되는 경우가 많다.
가장 잘 알려진 RNN의 단점으로는 기울기 폭발 또는 기울기 소실의 문제가 있다. 특히 기울기 소실의 문제가 나타나는 경우가 많은데, 이러한 기울기 소실은 역전파 단계에서 활성화 함수를 통해 발생한다. 이러한 기울기 소실이 발생하는 이유는 역전파하는 과정에서 생겨난다. 은닉층의 활성화 함수로 많이 사용하는 시그모이드 함수나 하이퍼볼릭 탄젠트 함수의 도함수는 1이하의 양의 실수를 값을 가지므로, 이를 반복적으로 곱하면, 0에 수렴하기 때문에 먼 기간의 정보가 출력 층과 먼 값은 거의 영향을 받지 않게 된다.
이러한 RNN의 기울기 소실 문제를 해결하기 위해 LSTM(Long Short-Term Memory)을 사용하기도 한다.

1-3-2. LSTM

LSTM(Long short-term memory)은 인공 신경망 중의 하나로, RNN((Recurrent Neural Network)의 기법 중 하나로 기존의 RNN의 기울기 소실 문제를 해결하기 위해 사용한다.
기존의 RNN과 LSTM의 차이점은 LSTM에는 망각 게이트(forget gate)가 있어서 과거의 정보 중에서 어떠한 값을 더 기억하고 있을지 망각을 할지 고르는 역할을 한다. 이렇게 망각 게이트를 통과한 값들은 셀 상태에서 입력 게이트에서 연산 된 값과 함께 다음 출력 게이트와 은닉 상태로 전달이 된다. 따라서 LSTM은 이러한 특징을 가지고 있어서 RNN에 비해 기울기 소실 문제가 일부 해결된다.

2. 연구 과정

2-1. 실험 환경

실험은 CPU는 8코어 16쓰레드의 AMD社의 Ryzen7 5700X, GPU는 RTX 3080, 메모리는 16GB의 하드웨어 환경에서 진행했다.
아래는 하드웨어와 소프트웨어 정보를 보여주는 표이다.

하드웨어	항목

CPU	AMD Ryzen 7 5700X
GPU	RTX 3080 10GB
RAM	DDR4 16GB

소프트웨어	버전
OS	Ubuntu 22.04.2 LTS
CUDA	12.0
Python	3.10.6
Pytorch	1.13.1+cu117
numpy	1.24.2
pandas	1.5.3
matplotlib	3.7.0
scikit-learn	1.2.1

2-2. 데이터 수집

사용한 항목	데이터 기간	출처
네이버에서 “코로나” 검색량	2020. 1. 24 ~ 2022. 12. 29.	네이버 데이터랩 (https://datalab.naver.com/)
평균 기온(°C)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
일사량(MJ/㎡)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
지면온도(°C)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
30cm 지중온도(°C)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
5m 지중온도(°C)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
미세먼지 농도(㎍/㎥)	2020. 1. 24 ~ 2022. 12. 29.	기상자료개방포털 (https://data.kma.go.kr/)
지하철 승객 수	2020. 1. 24 ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
서울시 코로나19 확진자 수	2020. 2. 5. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
추가 백신 1차 접종량	2021. 4. 21. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
누적 백신 1차 접종량	2021. 4. 21. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
추가 백신 2차 접종량	2021. 4. 21. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
누적 백신 2차 접종량	2021. 4. 21. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
추가 백신 3차 접종량	2021. 10. 13. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)
누적 백신 3차 접종량	2021. 10. 13. ~ 2022. 12. 29.	서울 열린데이터광장 (https://data.seoul.go.kr/)

데이터 파일을 구성하기 위해 논문에서 밝힌 출처대로 데이터를 수집하였다.

같은 변수를 사용하고자 하였으나 논문의 변수와 일부 차이가 있다. 차이가 있는 변수는 사회적 거리두기 단계, 백신 3차 접종 관련 데이터이다.

사회적 거리두기 단계의 경우에는 논문에 출처의 사이트가 폐쇄되었고, 다른 기관에서라도 제공하는 데이터를 확보하기 어려움이 있어서 이 실험에서는 변수 목록에서 제외하였다.

백신 3차 접종 관련 데이터는 논문에서 나타난 데이터의 기간보다 1년 이상의 데이터를 추가적으로 확보 할 수 있었고, 이 기간 동안 백신 3차 접종이 이루어졌기 때문에 성능 개선의 여지가 있다고 판단하여, 이 데이터를 사용하였다.

데이터의 기간은 2020년 2월 5일부터 2022년 12월 29일까지로 지정하였다. 그 이유는 서울시 열린 데이터 광장에서 공개된 정보 중에서 확진자 수 관련 정보가 2월 5일부터 있기 때문이다.

논문에서는 2021년 8월 3일까지의 데이터를 사용했지만, 현 시점에서 추가적으로 데이터를 구할 수 있어서 2022년 12월 31일까지의 데이터를 사용하였다.

그래서 논문과 다르게 3차 접종의 데이터를 추가적으로 활용할 수 있었다. 2020년 2월 5일부터 2022년 12월 29일까지의 기간을 '긴 기간'으로 부르고, 논문과 비교를 위해 논문에서 사용한 데이터의 기간(2020. 2. 5. ~ 2021. 8. 3.)을 '짧은 기간'으로 부를 것이다. 이렇게 서로 다른 두 기간을 가지고 실험을 진행할 예정이다.

2-3. 데이터 전처리

각 데이터 파일은 csv 파일의 형태로 다운로드 받았다. 이렇게 받은 여러 데이터 파일을 학습에 사용하기 위해 하나의 파일로 통합하는 과정을 거쳤다.

각 데이터 파일의 인코딩이 서로 달랐기 때문에 모든 파일의 인코딩을 'UTF-8'로 일치시켰다.

그리고 각 변수들을 연결하기 위해 날짜를 일치시켜야 했다. 날짜 형식이 'YYYY-MM-DD', 'YY.MM.DD', 'YYYYMMDD'로 파일마다 다 다르므로 'YYYYMMDD'의 정수 형태로 변환하여 대소비교로 시점 간의 비교가 편하도록 하였다.

또한 지하철 승객수 데이터의 원본은 아래의 그림과 같이 각 역마다 승객수가 다르다.

사용일자	노선명	역명	승차총승객수	하차총승객수
20200101	1호선	종각	20427	16301
20200101	1호선	시청	12126	10516
20200101	우이신설선	신설동	892	828
20200101	우이신설선	보문	917	855
20200101	우이신설선	성신여대입	2010	2363
20200101	우이신설선	정릉	2096	1989

여기에서 필요한 승객수를 계산하기 위해 '사용일자'열이 같은 경우를 묶어서 아래의 그림과 같이 승객수를 다시 계산하였다.

	일시	승차	하차
0	2020-01-01	3479698	3453691
1	2020-01-02	7680892	7656378
2	2020-01-03	8222729	8196739

승차승객수와 하차승객수가 근소하게 다르지만, 이중 승차승객수만 승객수로 사용하였다.

2-4. 이슈 및 해결 방법

문제가 있는 데이터 항목	문제 사항	해결방법
사회적 거리두기 단계	데이터 수집 불가	실험에서 누락
지하철 승객수	승차객수와 하차객수의 불일치	승차객수를 지하철 승객수로 사용
미세먼지 농도	일부 값 누락	시간별 데이터를 통해 일별 평균 계산

지하 온도	일부 값 누락	당일의 지면의 온도와 일치
기타	기타 누락	선형보간 (전날과 다음날의 평균의 값으로 계산 전날과 다음날이 없는 경우, 가장 가까운 날의 등차수열의 형태로 계산)

위에서 언급된 사회적 거리두기 단계와 지하철 승객수의 문제와 해결 방법은 데이터 전처리 과정에서 설명하였다.

중간에 누락된 값이 발생하면 데이터 프레임에서 Null 값으로 채워지고 학습 과정에서 loss 값을 계산할 수 없어서 tensor에는 NaN 값으로 저장된다. 이러한 문제를 해결하기 위해 누락된 값은 적절한 값으로 채우는 과정이 필요하다.

지중온도는 당일의 지표면 온도와 유사한 경우가 많으므로 당일의 지표면 온도로 채웠다.

미세먼지의 경우에는 일별 자료에는 없는 값들이 시간별 자료에는 있는 경우가 있다. 이러한 시간별 자료의 일간 평균치를 채워 넣었다. 이렇게 시간별 자료로 계산된 값들은 일별 자료의 값과 1~3정도의 작은 차이를 보여서 이러한 방식으로 채우는 것이 무리가 없다고 판단하였다.

백신 접종의 경우에는 데이터가 2021년 4월 21일부터 있었다. 그 이전의 데이터를 처리하기 위해 자료 제공처에 직접 문의하였으나, 그 이전의 자료는 없다는 답변을 받았다. 그래서 다른 방법으로 문제해결을 시도하였다. 실제로 우리나라의 백신접종은 2021년 2월 26일부터 시작했다는 기사를 바탕으로 2021년 2월 25일의 값을 0으로 놓고 4월 21일까지의 값을 동일하게 증가하는 등차수열의 형태로 채워 넣었다. 예를 들어 [9, NaN, NaN, NaN, 25]라면 [9, 13, 17, 21, 25]의 값으로 채워 넣는 것이다. 2차 접종의 경우 1차 접종과 4주의 간격을 두어 시작한 것으로 보고 이전의 방식대로 채워 넣었다. 그 이후의 비교적 최근인 2022년 11월 이후에는 주말마다 값이 비어 있는 것으로 보아 주말에는 접종하지 않은 것으로 보고 0으로 채워 넣었다. 그 이외의 간헐적으로 누락된 소수의 데이터는 전날과 다음날의 데이터의 평균으로 채워 넣었다.

3. 실험을 위한 데이터 셋 구성

3-1. 정규화

각 데이터는 Scikit-Learn의 MinMaxScaler() 함수로 0에서 1의 값으로 정규화 하였다. MinMaxScaler() 함수는 아래의 수식대로 최솟값을 0, 최댓값을 1로 바꿀 수 있다.

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$

이렇게 정규화를 하는 이유는 각 feature마다 다른 단위와 범위를 가지므로 혹여나 큰 범위를 가지는 변수의 영향력이 과대평가되는 것을 방지하기 위함이다.
그리고 이런 정규화 과정은 Train set과 Test set을 나누기 전에 해야 한다. 그 이유는 만약 Train set의 범위와 Test set의 범위가 크게 차이 나는 경우 제대로 된 학습과 예측이 어려워지기 때문이다.

인공 신경망에는 텐서의 형태로 입력을 하여 학습을 시킬 수 있는데 이러한 텐서를 쉽게 만들기 위해 먼저 데이터셋을 만들어야 한다. 데이터셋은 5일치의 feature를 가지고 특정 시기의 예측할 값을 구성해야 한다. 그 예시는 아래의 그림과 같다. 이는 numpy array의 형태로 저장된다.

3-2. split, 저자의 데이터와 Dtype

3-2-1. Train Test Split

훈련 데이터와 테스트 데이터는 8:2의 비율로 나눈다.

긴 기간에서는 각 변수별로 1063일치의 데이터가 있는데 이의 80%인 850일째인 2022년 6월 1일을 기준으로 나눈다. 하지만 논문에서 소개한 sliding window에 따라서 테스트 셋에서 입력을 위해 이전 5일치의 데이터가 필요하다. 따라서 일부 겹치는 부분이 발생한다. 엄밀하게는 7, 14일 후의 추가 확진자 수 예측을 하는 데에는 이전 5일치뿐만 아니라 더 이전의 데이터도 필요하지만, 편의상이 부분은 생략되었다.

아래의 표는 Train set과 Test set에서의 데이터가 사용된 기간을 나타낸다.

	Train set	Test set
긴 기간	2020. 2. 5. ~ 2022. 6. 1.	2022. 5. 28. ~ 2022. 12. 29.
짧은 기간	2020. 2. 5. ~ 2021. 4. 16.	2021. 4. 12. ~ 2021. 8. 3.

3-2-2. 저자의 데이터와 Dtype

운이 좋게 저자의 github에서 논문에 사용하기 위해 구성한 코드와 데이터를 찾을 수 있었다. 논문과는 다른 부분이 존재하지만 이와 유사하게 논문의 실험을 구성한다고 판단할 수 있었다. 저자의 github에서 확보한 데이터인 “tr.csv”를 직접 사용하면 데이터 셋을 구성할 때, numpy.array 안에 일부 값이 numpy.object_ 형태로 저장되어 텐서로의 변환이 불가능하다. 이때, dtype의 parameter 값을 주어 텐서로의 변환이 가능하게 만들 수 있다.

4. 실험 구성

4-1. 실험 과정

- 논문의 변수와 맞게 직접 데이터 파일을 구성하였다.
  이 파일은 2020년 2월 5일 ~ 2022년 12월 29일의 데이터를 가지고 있다.
  이 데이터 파일은 df.csv이다.
- 논문에 따르면, 2020년 1월 24일 ~ 2021년 8월 3일의 데이터를 사용한다.
  논문과의 비교를 위해 2020년 2월 5일 ~ 2021년 8월 3일의 데이터 파일을 생성하였다.
  이 데이터 파일은 df_short.csv이다.
- 논문 저자의 github에서 직접 코드와 데이터(tr.csv)를 확인할 수 있었는데 저자가 만든 데이터를 사용하여 직접 실험을 진행하였다.
  이 데이터 파일은 tr.csv이다.

1. df의 RNN
2. df의 LSTM
3. df_short의 RNN
4. df_short의 LSTM
5. tr의 RNN
6. tr의 LSTM

각각의 실험 번호별로 1, 7, 14일 후의 서울시의 코로나19 추가 확진자 수 예측을 하는 실험을 진행했다.

4-2. 파일 이름

4-2-1. 코드

1~6번 실험은 1, 7, 14일로 구분하였으며, RNN과 LSTM 사용 여부는 파일 제목의 가장 앞에 지정하였다.

	1일	7일	14일
1번	rnn_1	rnn_7	rnn_14
2번	lstm_1	lstm_7	lstm_14
3번	rnn_short_1	rnn_short_7	rnn_short_14
4번	lstm_short_1	lstm_short_7	lstm_short_14
5번	rnn_tr_1	rnn_tr_7	rnn_tr_14
6번	lstm_tr_1	lstm_tr_7	lstm_tr_14

4-2-2. 데이터

1번	2번	3번
df.csv	df_short.csv	tr.csv

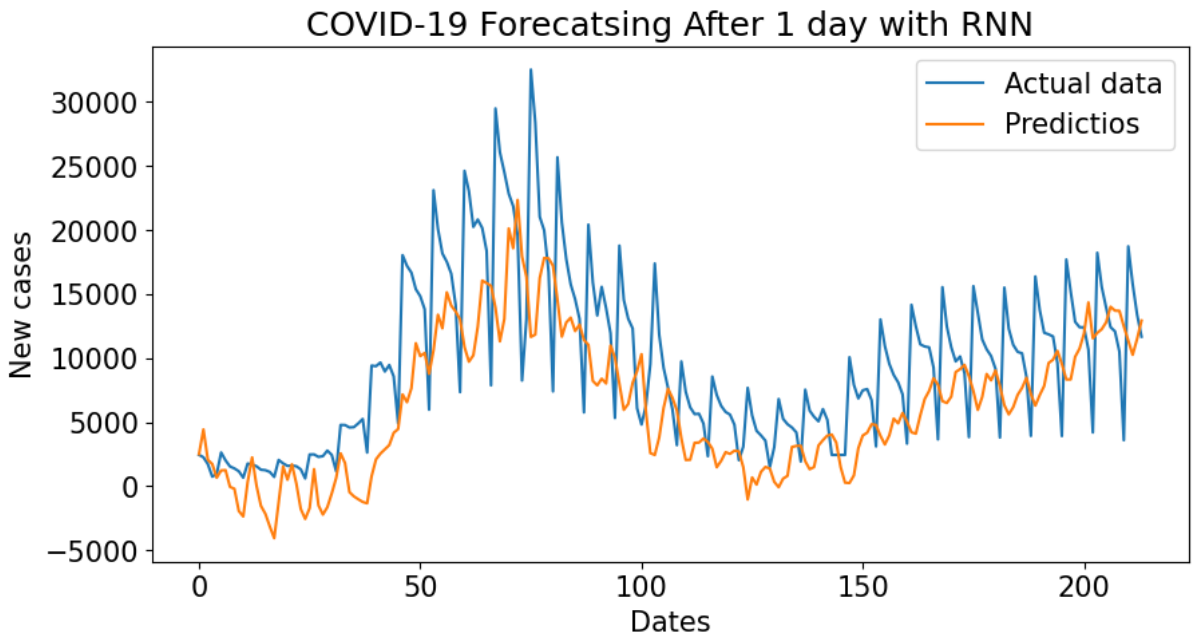
5. 연구 결과

각 실험별로 5회씩 반복하여 RMSE 값의 평균을 구한다.

5-1. df의 RNN

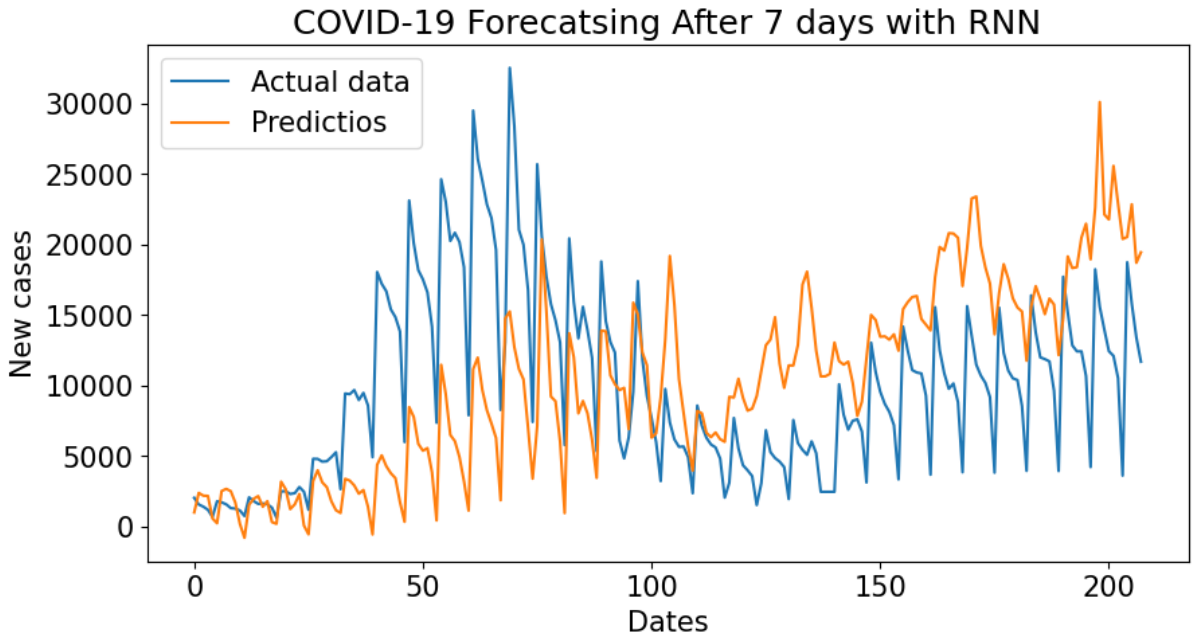
- 은닉층 개수 : 64
- epoch : 3000

5-1-1. 1일 후 예측



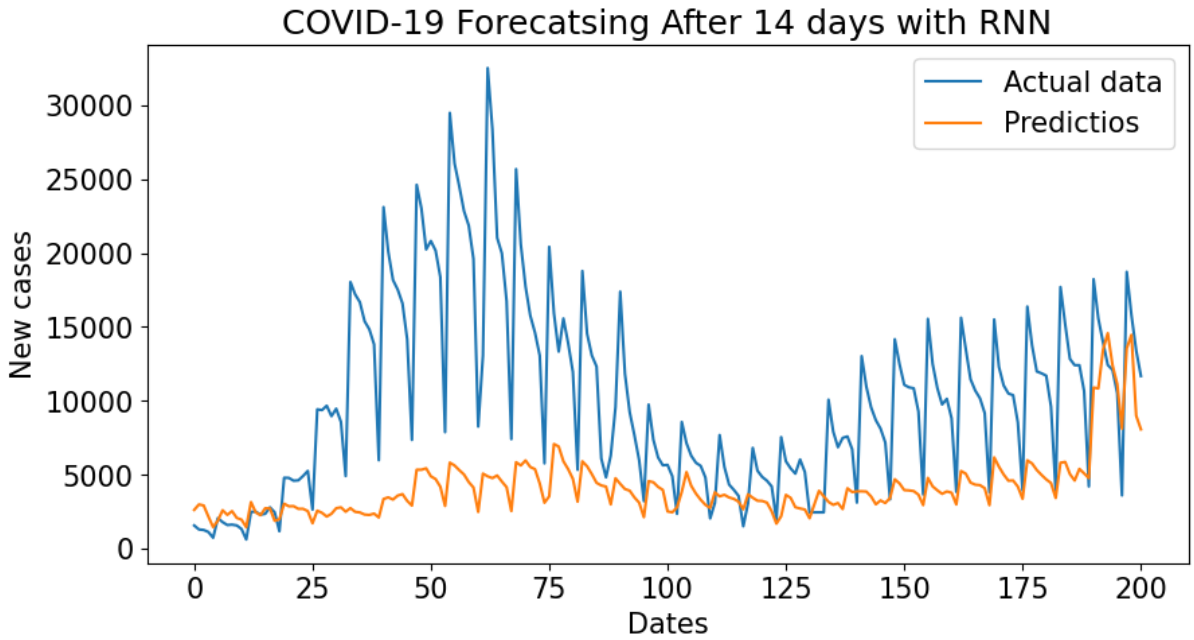
1. 5033.792660492766
2. 13380.461688915871
3. 6345.237836276129
4. 5804.256881194359
5. 5717.110863174513

5-1-2. 7일 후 예측



1. 7662.672381201455
2. 7615.3163599330255
3. 6017.581867316935
4. 8672.970110328004
5. 6424.0065188121

5-1-3. 14일 후 예측



1. 8317.579217824392
2. 9917.584772349312
3. 16932.369527018163

4. 5658.594642702561
5. 16073.748056540353

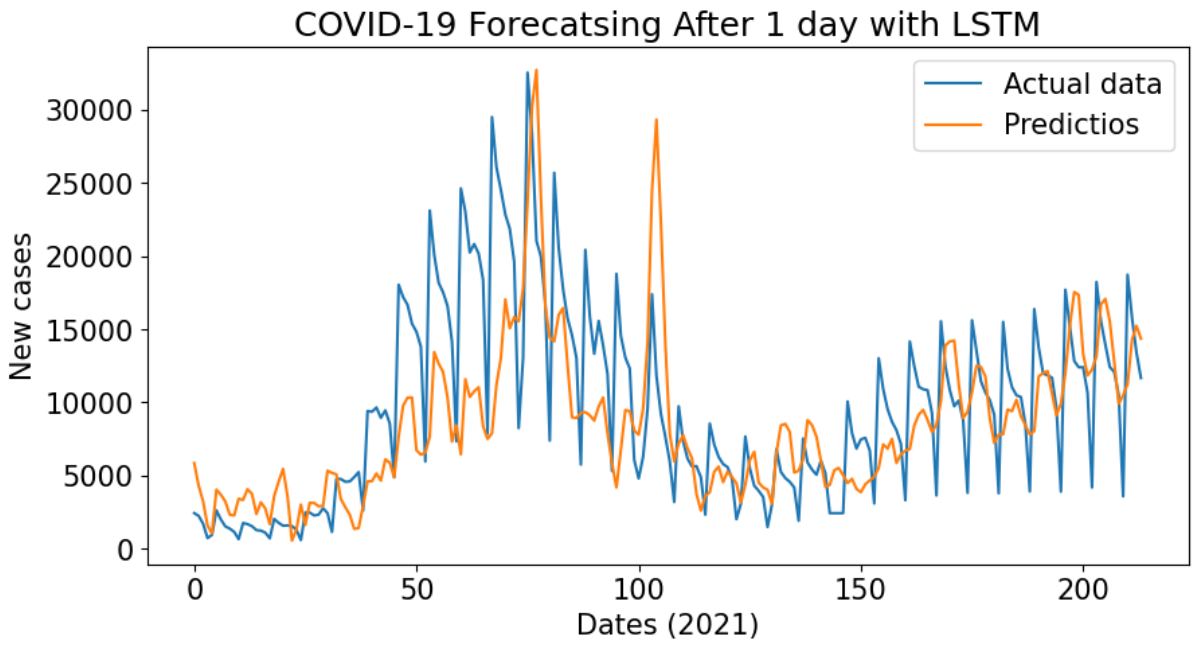
5-1-4. 요약

	1일 후	7일 후	14일 후
1회차	5034	7663	8318
2회차	13380	7615	9918
3회차	6345	6018	16932
4회차	5804	8673	5658
5회차	5717	6424	16074
평균	7256	7279	11380

5-2. df의 LSTM

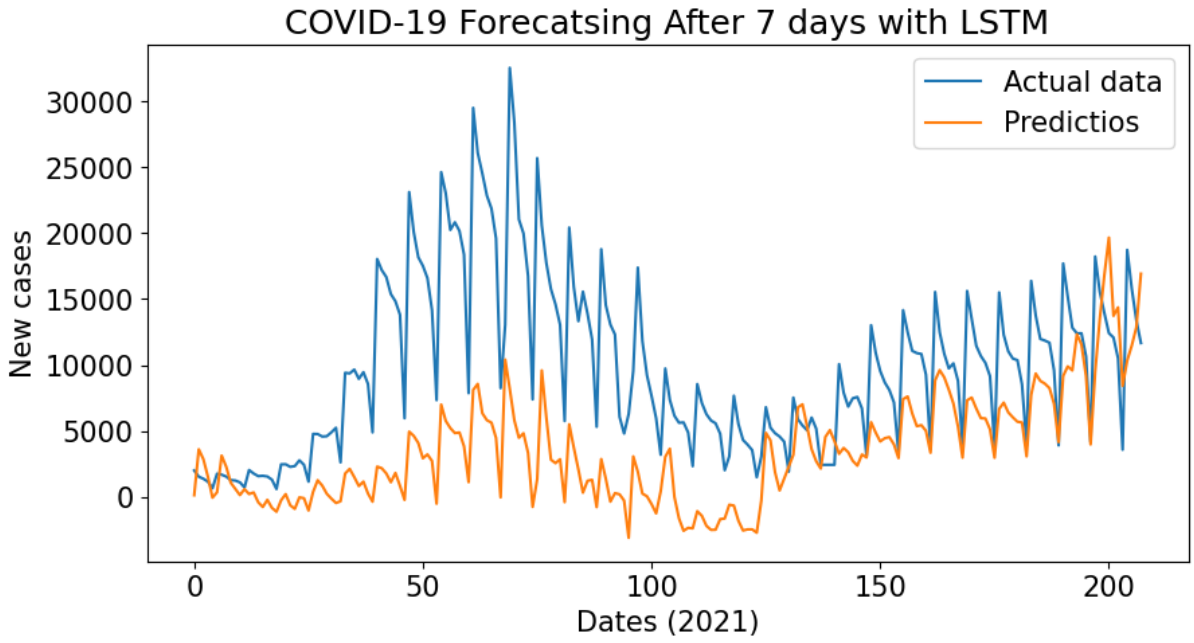
- 은닉층 개수 : 5
- epoch : 3000

5-2-1. 1일 후 예측



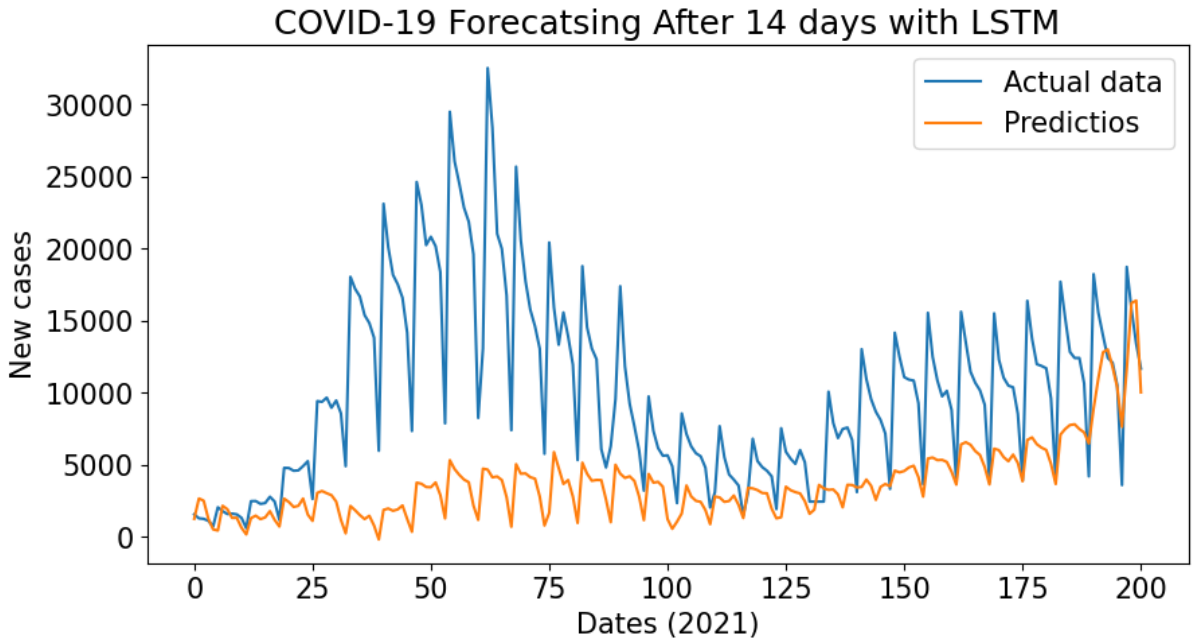
1. 5143.2281938037595
2. 6318.514788726989
3. 5227.249665928115
4. 8409.440176196555
5. 6520.492226617727

5-2-2. 7일 후 예측



1. 8506.329164034443
2. 6310.978236242044
3. 7289.686033483441
4. 7649.4195583239
5. 8956.53523662807

5-2-3. 14일 후 예측



1. 8728.601119568257
2. 9323.643790685237
3. 9736.846693962469

4. 9491.814782874095
5. 8622.864495060596

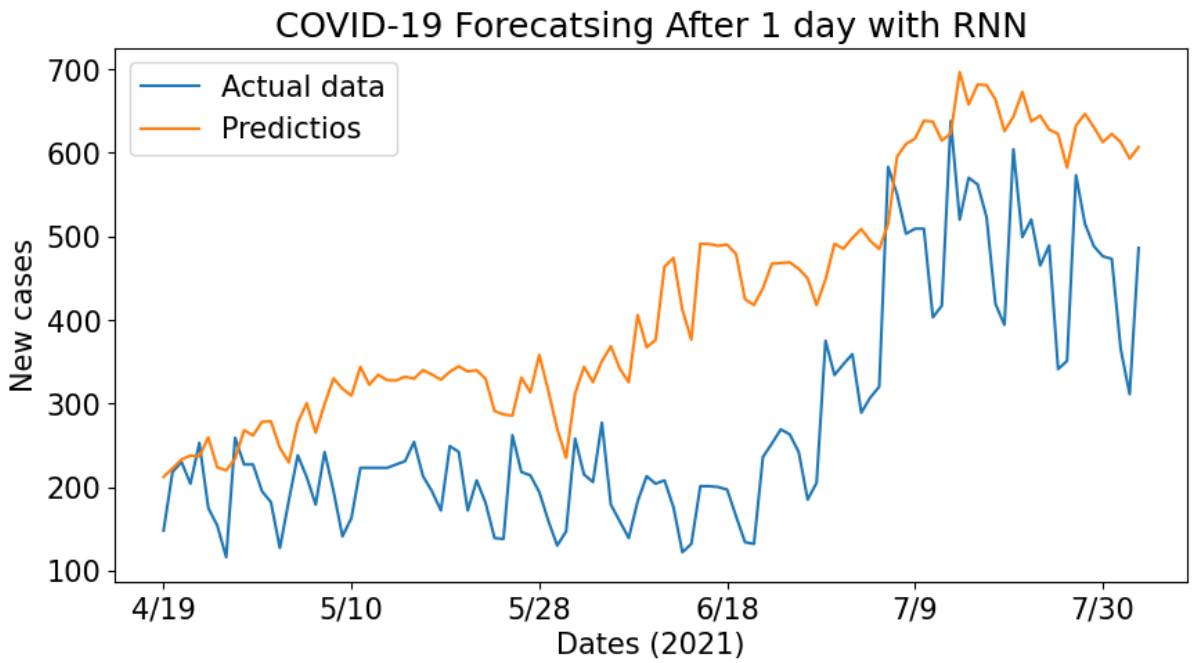
5-2-4. 요약

	1일 후	7일 후	14일 후
1회차	5143	8506	8729
2회차	6319	6311	9324
3회차	5227	7290	9737
4회차	8409	7649	9492
5회차	6520	8957	8623
평균	6323.6	7743	9181

5-3. df_short의 RNN

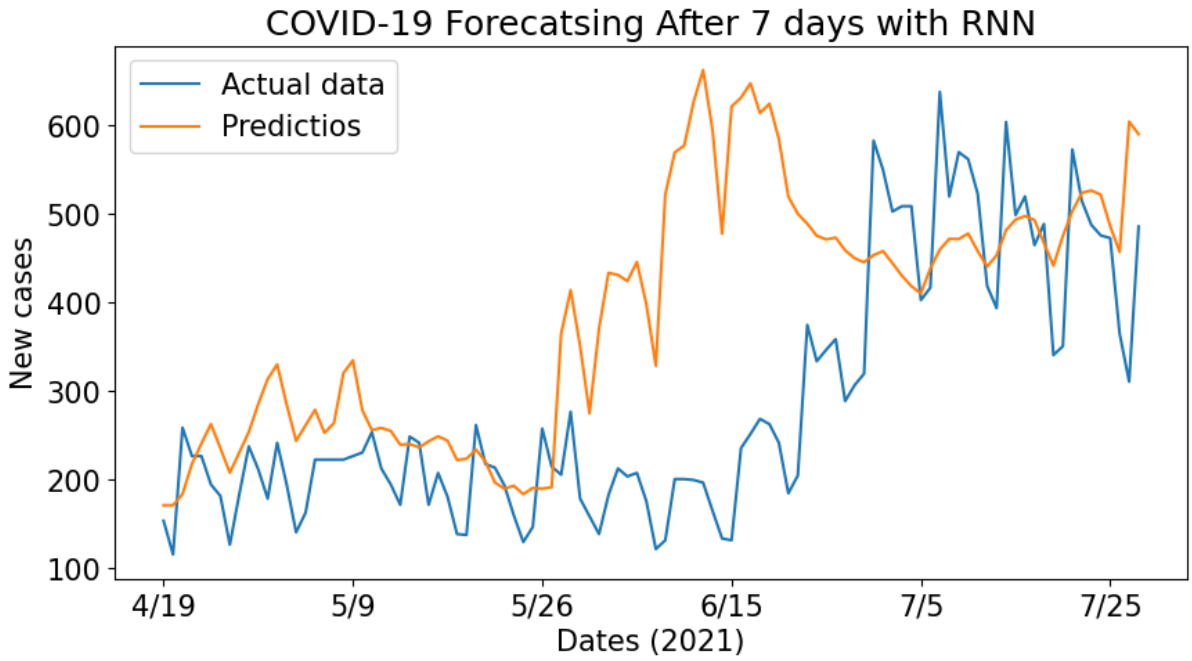
- 은닉층 개수 : 15
- epoch : 300

5-3-1. 1일 후 예측



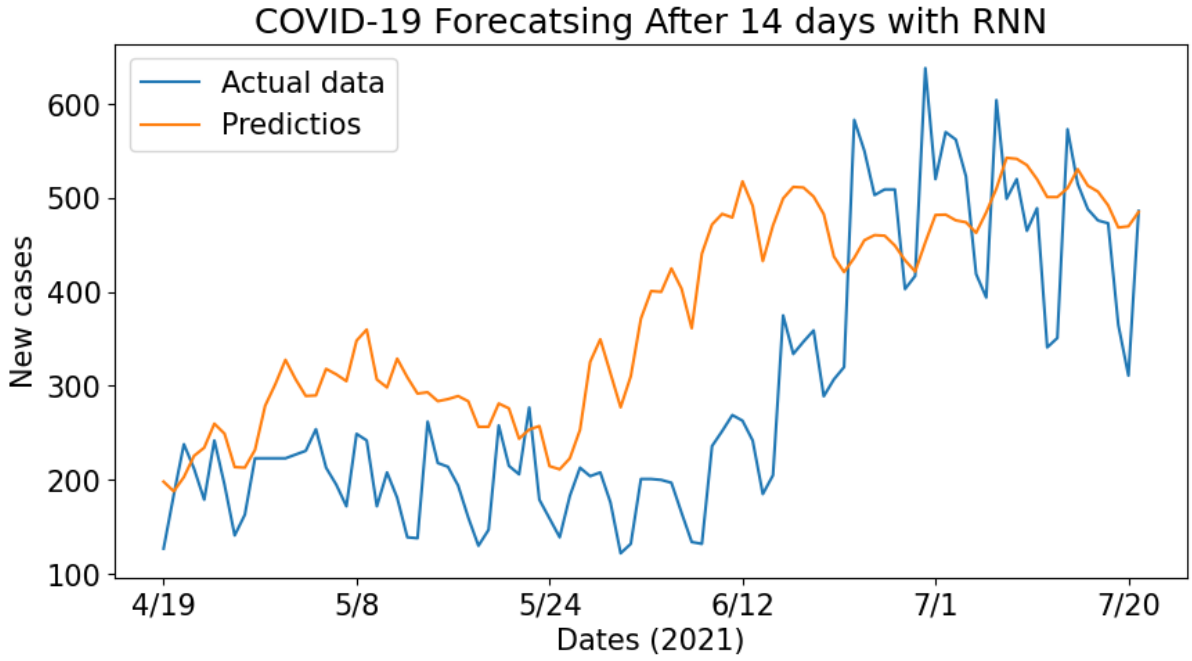
1. 164.52118156593434
2. 104.6804942070618
3. 214.26054910648534
4. 106.73195403861259
5. 104.49644431896527

5-3-2. 7일 후 예측



1. 176.7456895556227
2. 191.48904673897022
3. 183.9604759681171
4. 216.69353273307763
5. 189.85777081196946

5-3-3. 14일 후 예측



1. 126.87479471904238
2. 105.20065395675734

3. 108.67699130825655
4. 155.0758749413899
5. 167.71555324362788

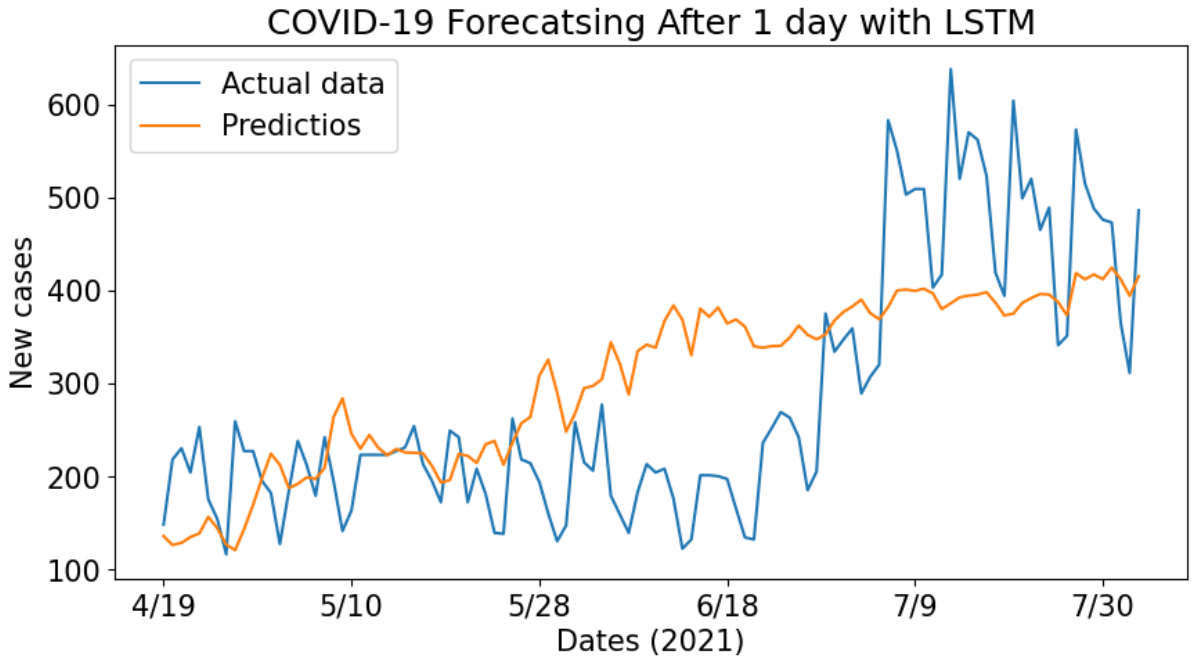
5-3-4. 요약

	1일 후	7일 후	14일 후
1회차	165	177	127
2회차	105	191	105
3회차	214	184	109
4회차	107	217	155
5회차	104	190	168
평균	139	191.8	132.8

5-4. df_short의 LSTM

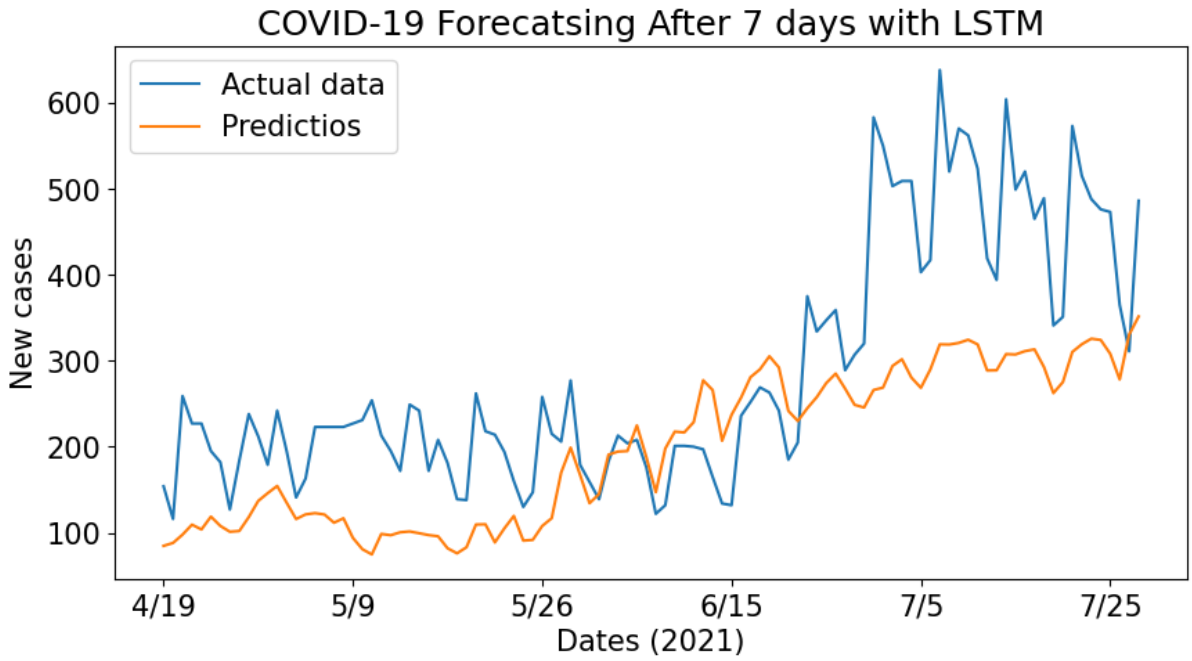
- 은닉층 개수 : 5
- epoch : 300

5-4-1. 1일 후 예측



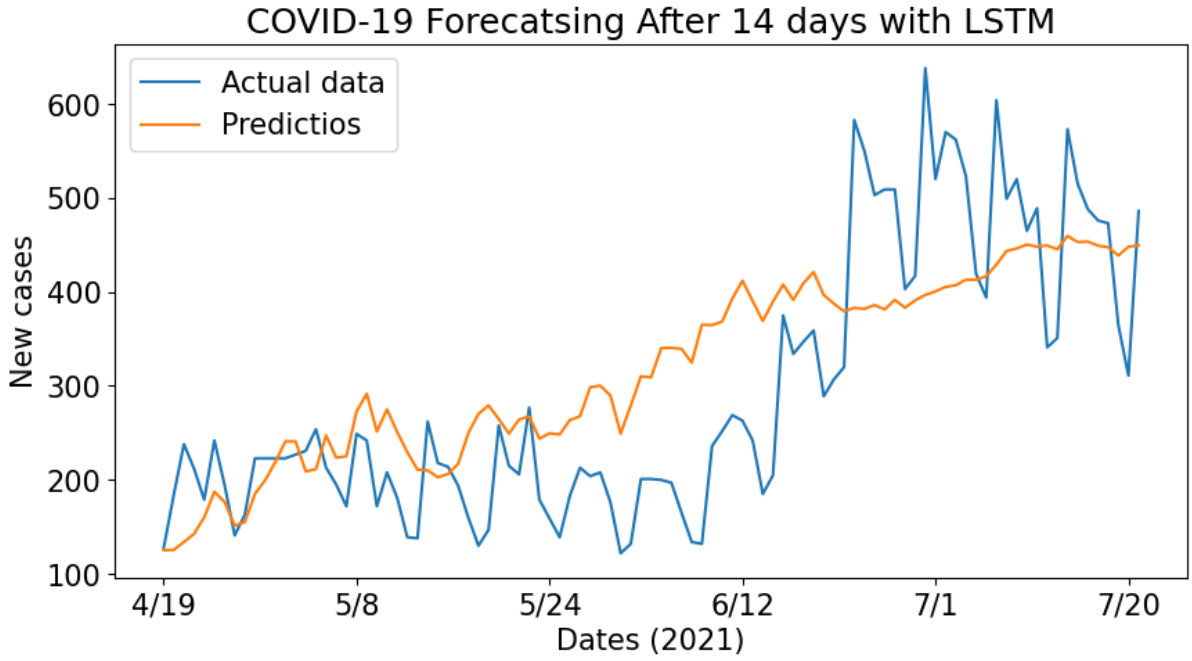
1. 109.52205009902512
2. 109.51542454896294
3. 142.559525204808
4. 115.10144207136506
5. 111.9310634036599

5-4-2. 7일 후 예측



1. 119.79463608507416
2. 216.0162632948308
3. 162.97371466370325
4. 119.90630093515192
5. 190.55331182663977

5-4-3. 14일 후 예측



1. 99.0926569354352
2. 122.58046433905022

3. 216.15350477370524
4. 117.99558282257014
5. 176.38373939109601

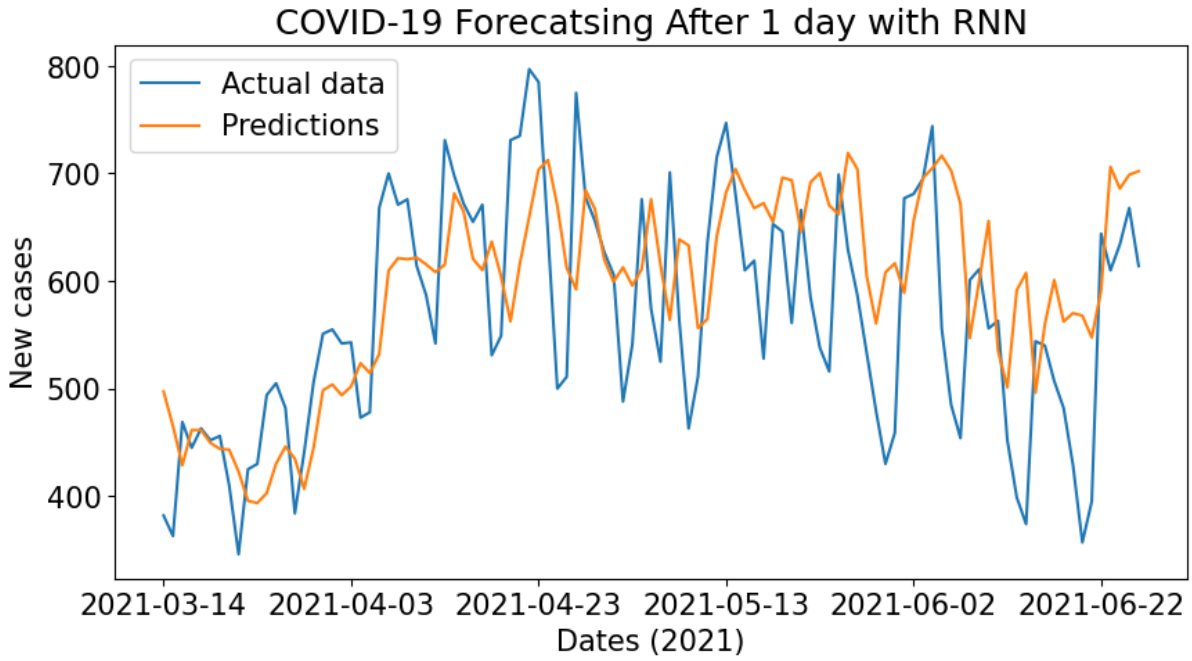
5-4-4. 요약

	1일 후	7일 후	14일 후
1회차	110	120	99
2회차	110	216	123
3회차	143	163	216
4회차	115	120	118
5회차	112	191	176
평균	118	162	146.4

5-5. tr의 RNN

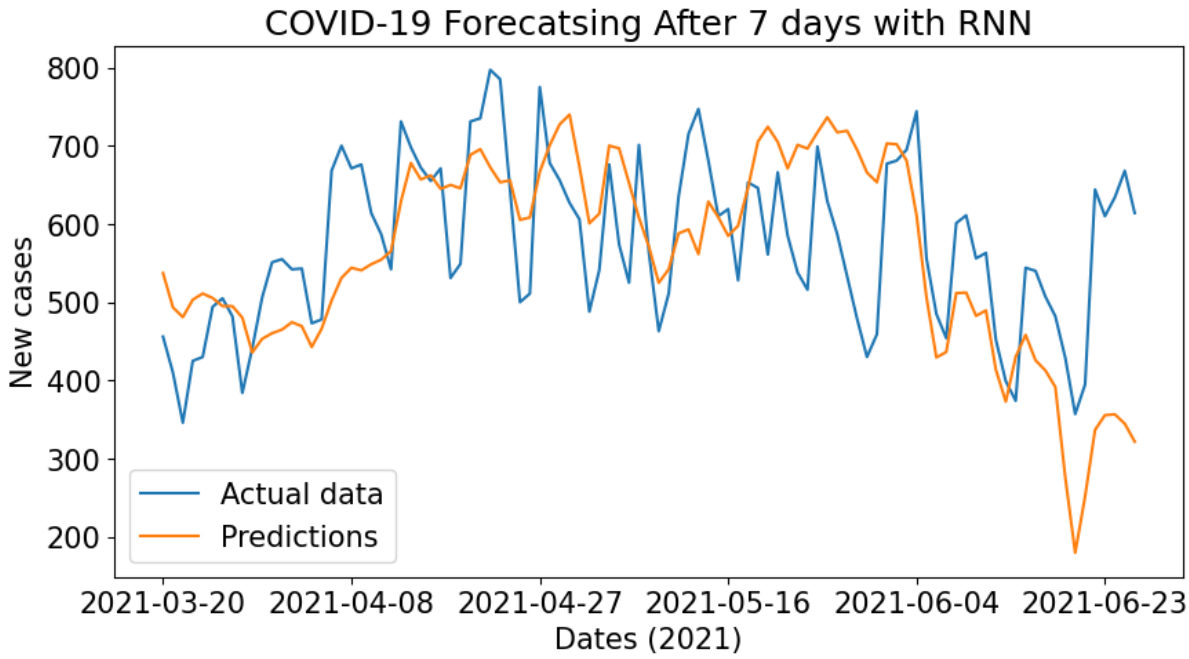
- 은닉층 개수 : 15
- epoch : 300

5-5-1. 1일 후 예측



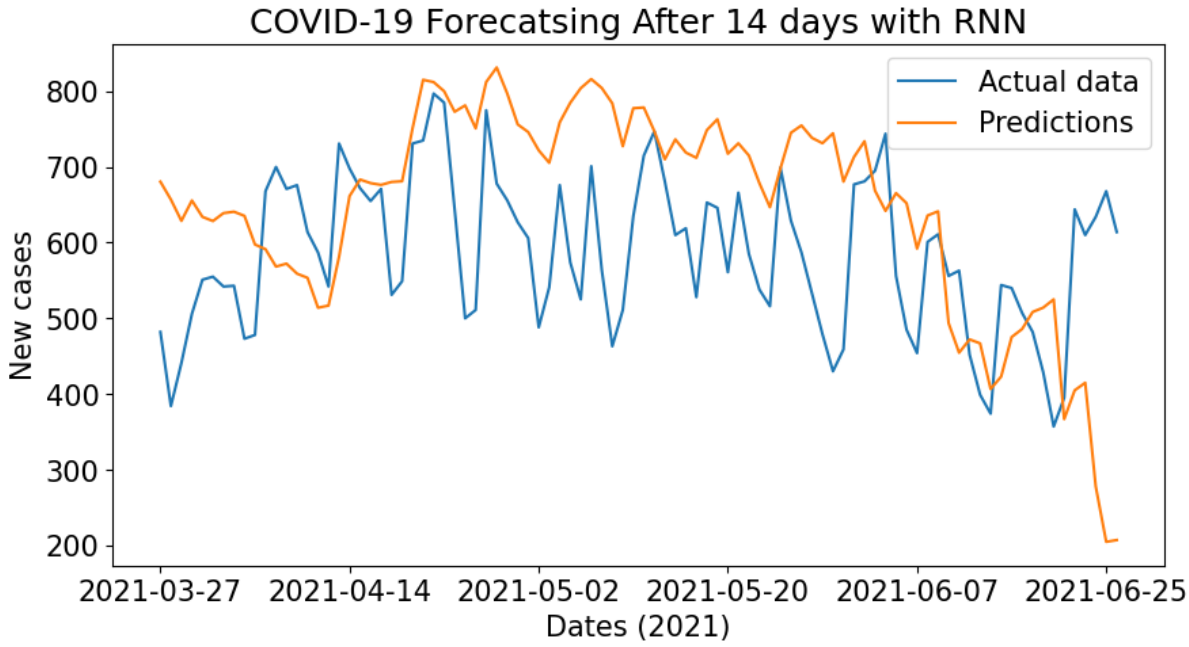
1. 96.03071463018478
2. 101.47715714952203
3. 132.5519294969235
4. 186.35088933287412
5. 116.63339504630156

5-5-2. 7일 후 예측



1. 115.90524238788099
2. 118.04890599388621
3. 102.47477277703112
4. 415.79462433919997
5. 199.4273892761327

5-5-3. 14일 후 예측



1. 157.45458156152242
2. 214.86768296185815
3. 189.77803598913468

4. 511.2696707878963
5. 206.48247646675867

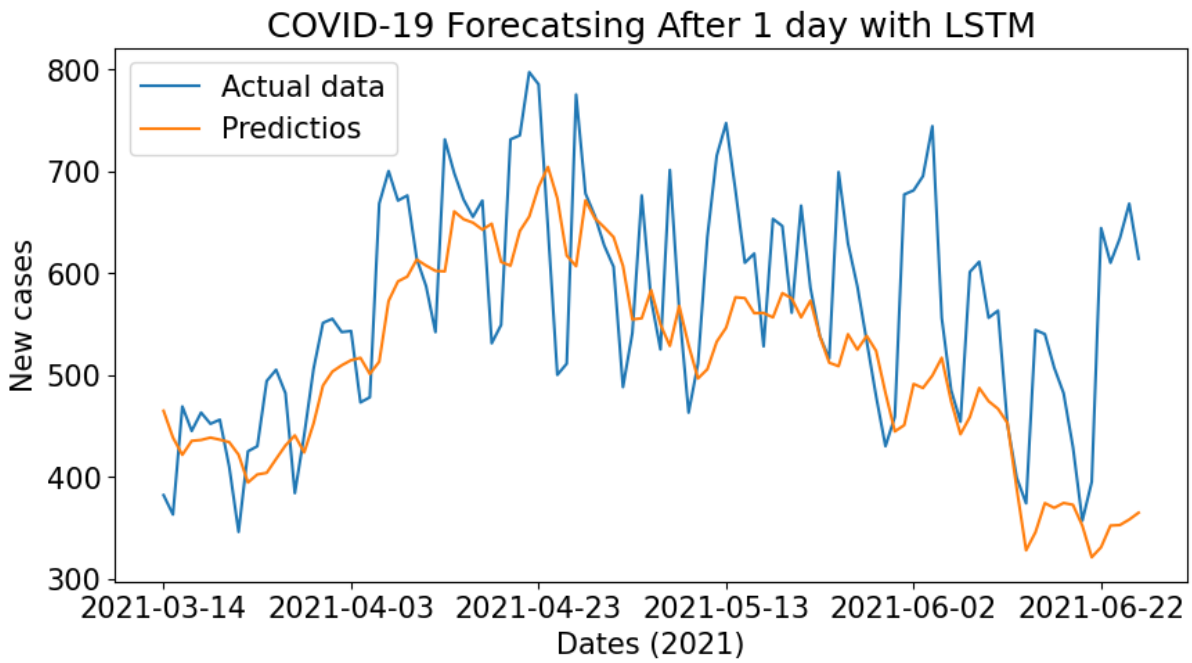
5-5-4. 요약

	1일 후	7일 후	14일 후
1회차	96	116	157
2회차	101	118	215
3회차	133	102	190
4회차	186	416	511
5회차	117	199	206
평균	126.6	190.2	255.8

5-6. tr의 LSTM

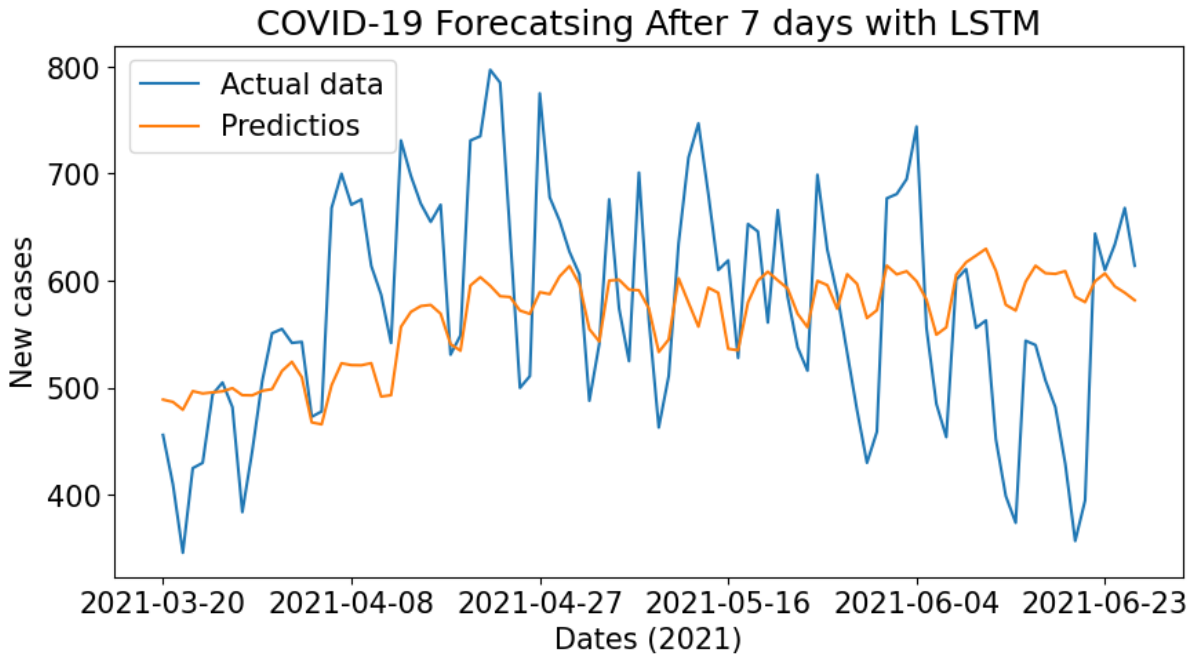
- 은닉층 개수 : 15
- epoch : 300

5-6-1. 1일 후 예측



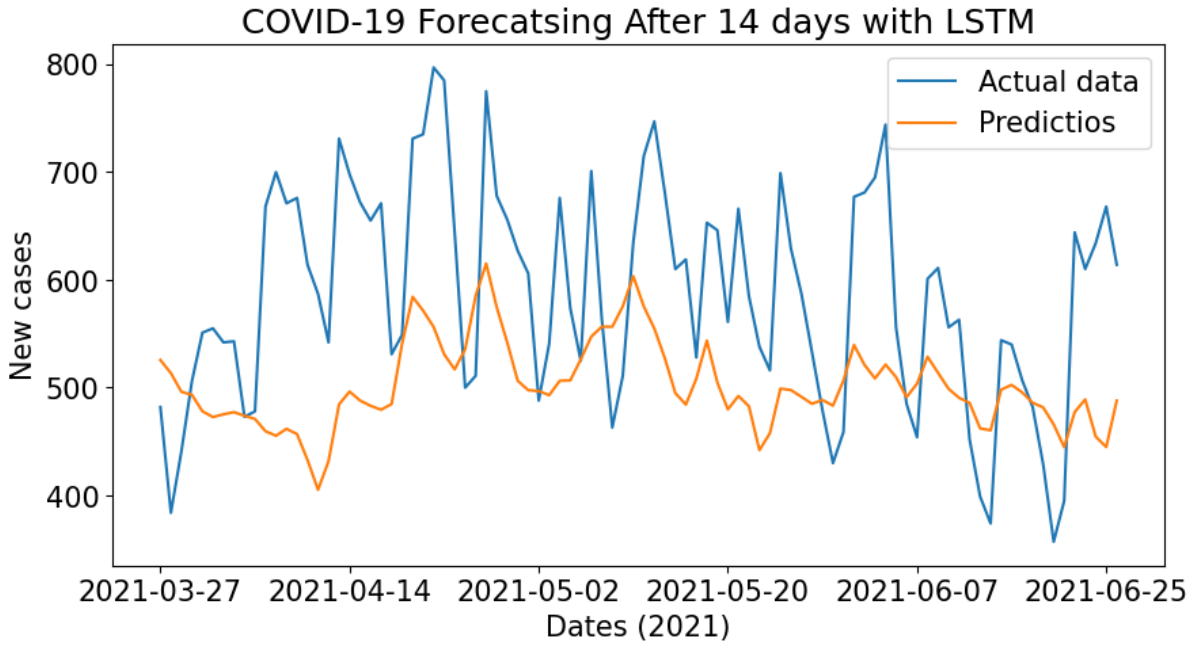
1. 110.68108684912899
2. 188.86572985204378
3. 162.5866034277949
4. 176.31198521750986
5. 101.03484067823166

5-6-2. 7일 후 예측



1. 164.59365983740943
2. 96.92991819516071
3. 106.05641579432941
4. 260.5228602488795
5. 234.34530820928737

5-6-3. 14일 후 예측



1. 126.37735172782995
2. 266.94672011496584
3. 96.85523002915602

4. 89.83613668918908
5. 137.43874344490948

5-6-4. 요약

	1일 후	7일 후	14일 후
1회차	111	165	126
2회차	189	97	266
3회차	163	106	97
4회차	176	260	90
5회차	101	234	137
평균	148	172.4	143.2

5-7. RNN과 LSTM의 비교

5-7-1. df의 RNN과 LSTM의 비교

	RNN, 1일 후	LSTM, 1일 후	RNN, 7일 후	LSTM, 7일 후	RNN, 14일 후	LSTM, 14일 후
1회차	5034	5143	7663	8506	8318	8729
2회차	13380	6319	7615	6311	9918	9324
3회차	6345	5227	6018	7290	16932	9737
4회차	5804	8409	8673	7649	5658	9492
5회차	5717	6520	6424	8957	16074	8623
평균	7256	6323.6	7279	7743	11380	9181

1, 14일 후 예측 시, RMSE 값이 LSTM에서 낮은 것을 확인할 수 있다.

5-7-2. df_short의 RNN과 LSTM의 비교

	RNN, 1일 후	LSTM, 1일 후	RNN, 7일 후	LSTM, 7일 후	RNN, 14일 후	LSTM, 14일 후
1회차	165	110	177	120	127	99
2회차	105	110	191	216	105	123
3회차	214	143	184	163	109	216
4회차	107	115	217	120	155	118
5회차	104	112	190	191	168	176
평균	139	118	191.8	162	132.8	146.4

1, 7일 후 예측 시, RMSE 값이 LSTM에서 낮은 것을 확인할 수 있다.

5-7-3. tr의 RNN과 LSTM의 비교

	RNN, 1일 후	LSTM, 1일 후	RNN, 7일 후	LSTM, 7일 후	RNN, 14일 후	LSTM, 14일 후
1회차	96	111	116	165	157	126
2회차	101	189	118	97	215	266
3회차	133	163	102	106	190	97
4회차	186	176	416	260	511	90
5회차	117	101	199	234	206	137
평균	126.6	148	190.2	172.4	255.8	143.2

1, 14일 후 예측 시, RMSE 값이 LSTM에서 낮은 것을 확인할 수 있다.

5-8. 논문과의 비교

	논문 RNN	논문 LSTM	df_short RNN	df_short LSTM	tr_RNN	tr_LSTM
1일 후	82	65	139	118	126.6	148
7일 후	105	83	191.8	162	190.2	172.4
14일 후	142	130	132.8	146.4	255.8	143.2

1일 후의 예측에서는 논문과 큰 차이를 보인다.

14일 후에서는 논문과 근소한 차이를 보인다.

6. 결론

6-1. RNN과 LSTM의 차이

RNN에 비해 LSTM의 성능이 확실하게 좋다라고 보기에는 어렵다.

다른 논문(RNN과 LSTM을 이용한 주가 예측율 향상을 위한 딥러닝 모델)에서도 RNN과 LSTM의 예측 오차율 향상 폭이 크지 않다.

6-2. 초매개변수(Hyperparameter)와 과적합(Overfitting)

초매개변수에 따라 결과가 크게 달라지는 것을 확인할 수 있었다. 특히 학습 반복 횟수인 epoch와 은닉층의 수의 영향을 가장 크게 받았다.

epoch를 크게 설정하면, train data에 대해 주어진 입력값에 대해 실제값과 가까운 예측값을 출력한다.
은닉층 수를 크게 설정하면, feature의 수가 많은 데이터의 복잡한 관계를 잘 학습할 수 있다.
그러나 과도하게 epoch과 은닉층 수를 크게 설정하면 train data에 대해 학습을 잘하는 것으로 보이지만 과적합 문제가 가장 쉽게 발생한다.
이러한 과적합 문제는 예측 성능을 크게 떨어뜨리는 결과로 나타난다.

은닉층 수가 적절하지 않으면, test data에서 예측값이 실제값에 비해 과도하게 크거나 음수값처럼 과도하게 작은 값이 나온다. 실제값은 일주일의 단위로 주기성을 가지고 있었는데, 예측값이 그 주기성을 따라가지 않고 평탄하게 작은 범위 내에서 나오는 경우도 있었다. 이는 손실함수로 MSE를 사용하였기 때문에 그냥 단순히 실제값의 변동 폭의 중간 정도의 값을 일정하게 예측하는 것이 손실함수를 최소화하는 방법으로 계산된 것으로 추정된다.

사용한 데이터에 따라 적절한 epoch 수가 다르다. 긴 기간(df)에서는 3000회의 epoch에서도 문제가 없었다. 하지만, 논문과의 비교를 위한 짧은 기간(df_short)에서는 3000회의 epoch는 과도하게 많아서 과적합의 문제가 발생하는 것으로 보아 그의 10분의 1의 수준인 300회의 epoch가 적절한 것으로 판단된다.

여러번 epoch와 은닉층의 수를 바꿔가며 가장 나은 예측 성능을 나타내는 값으로 실험을 하였다. 그 값들은 데이터와 RNN과 LSTM 중 어떤 모델을 사용하였는 가에 따라서도 달라진다.

6-3. 데이터의 문제

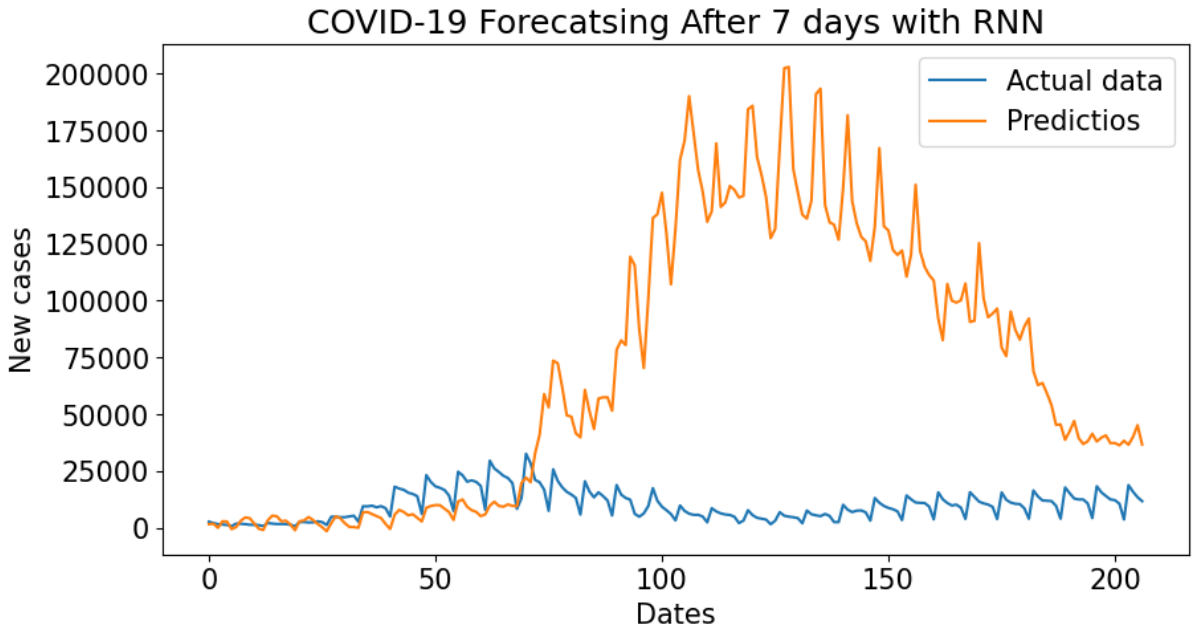
6-3-1. 불필요한 feature

긴 기간(df)의 데이터에서 백신 3차 접종관련 feature들을 포함하고 있다. 백신 3차 접종기간이 짧고 초반에는 0의 수치를 나타낸다. train data에서는 0값을 가지는 부분이 많고, test data에서는 0이 아닌 값을 가지는 부분이 많다. 따라서 이 백신 3차 접종관련 feature들은 학습 과정에서 과소 혹은 과대 평가 되었을 가능성이 있다.

6-3-2. 날짜 설정의 문제

처음에 데이터 파일을 만들 때, 여러 데이터에서 날짜가 일치하는 항목끼리 합쳐 하나의 데이터 파일을 구성하였다. 그러나 이 과정에서 일별 코로나 추가 확진자 수에 관련된 날짜는 발표일 기준이다. 대부분 전날의 코로나 추가 확진자 수를 합산하여 그 다음날에 발표를 했다. 따라서 일별 코로나 추가 확진자 수에 대한 날짜가 실제 추가 확진자 발생과 하루의 차이를 보인다. 저자 데이터를 참고하면 발표일이 아닌 실제 확진자가 발생한 날짜를 기준으로 데이터를 구성하였다는 차이를 보이고 있었다.

아래의 그래프는 발표일을 기준으로 만든 데이터 파일이며, 예측 성능이 크게 떨어지는 것을 확인 할 수 있었다. 따라서 위의 실험결과는 이 문제를 실제 확진자 발생일을 기준으로 데이터를 수정하여 한 실험의 결과이다.



6-4. 성능 평가의 문제

1. 저자 데이터는 전국 단위의 데이터이다. 따라서 이 실험에서 비교가 적절하지 않다.
2. 2022년 데이터까지 포함한 경우 총 확진자 수의 범위가 다르다.
   최대 값이 128375, 638으로 다르다.
   → RMSE값을 위의 최대 값으로 나누어 주면 긴 기간(df)의 값이 더 작게 나올 것이다.