[회귀 복습]

- **서울시 평균 기온 예측 해커톤**
  - 서울시의 평균기온을 예측하는 인공지능 모델을 개발하는 것
  - 제공된 데이터를 통해 기온 및 일교차, 강수량, 평균습도 등 다양한 날씨 데이터를 분석
  - https://dacon.io/competitions/official/236200/overview/description

RNN(Recurrent Neural Network)

1. RNN

순환 신경망

- 입력과 출력을 시퀀스 단위로 처리

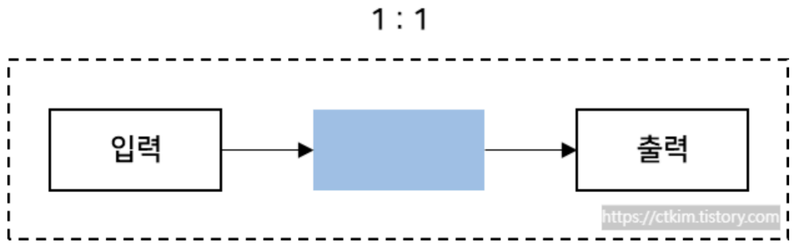
  - 시퀀스란 문장 같은 단어가 나열된 것
  - 이러한 시퀀스들을 처리하기 위해 고안된 모델을 시퀀스 모델
  - 그중에서 RNN은 딥 러닝의 가장 기본적인 시퀀스 모델

- 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할을 하는 노드를 셀

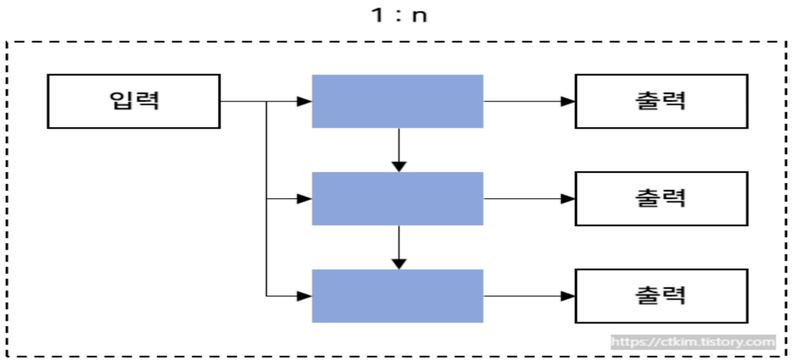
  - 이 셀은 이전의 값을 기억, 일종의 메모리 역할을 수행(**메모리 셀)**
  - 은닉층의 메모리 셀에서 나온 값이 다음 은닉층의 메모리 셀에 입력 ➔이 값을 은닉 상태

1. RNN 형태

- **입력, 출력의 개수에 따라 형태 결정**
  - 일대일 형태 (Vanilla Neural Network)
    - 한개의 입력에 대한 한 개의 출력을 생성하는 모델
    - 1:1 RNN은 간단한 기계학습 문제에 사용



  - 일대다 형태
    - 한개의 입력에 여러개의 출력 생성
    - 시**계열 데이터의 예측, 감정분석, 번역등에 사용**



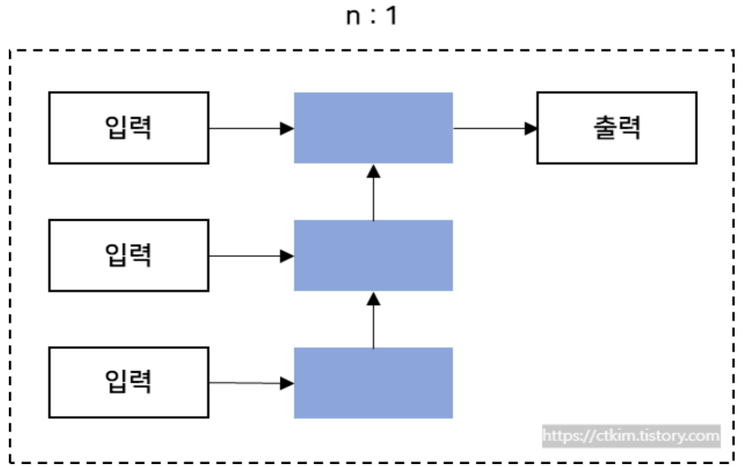
  - RNN은 단어의 어순에 따라 문장의 의미가 달라지고 앞에 어떤 단어가 쓰였는지 기억해야 뒤에. 오는 단어를 예측하는 등의 문제를 풀 때 주로 활용.

1. RNN 형태

- **입력, 출력의 개수에 따라 형태 결정**
  - 다대일 형태

    - 여러개 입력에 대한 한개의 출력을 생성하는 모델

    -N:1 형태의 RNN은 시계열 데이터의 예측, 상태감지, 경고 발생등에 사용

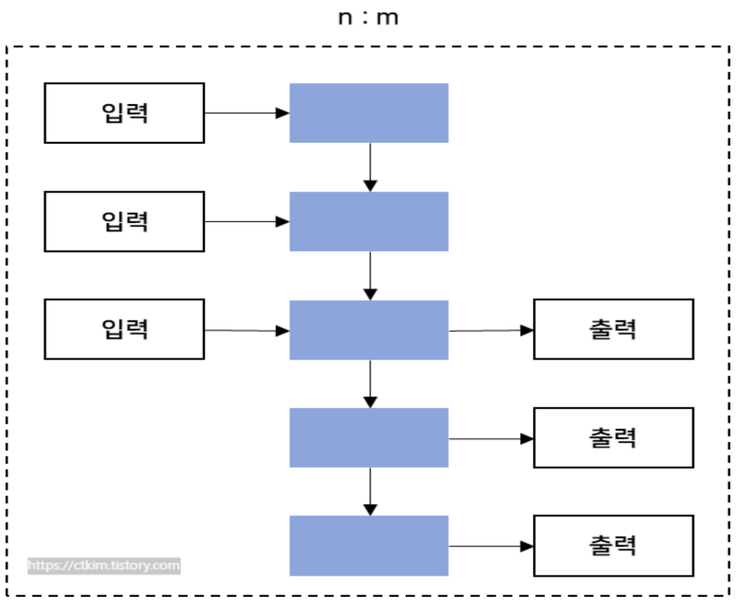


  - 다대다 형태

    - 여러개의 입력에 대해 여러개의 출력을 생성하는 모델

    - 복잡한 문제를 풀수있는 능력

    - 비디오분류, 이미지 캡셔닝, 게임API등 문제에 적용



1. CNN과 RNN

Neural Network 구조

CNN (Convolutional Neural Networks)

- (이미지와 같은) spatial data 기반

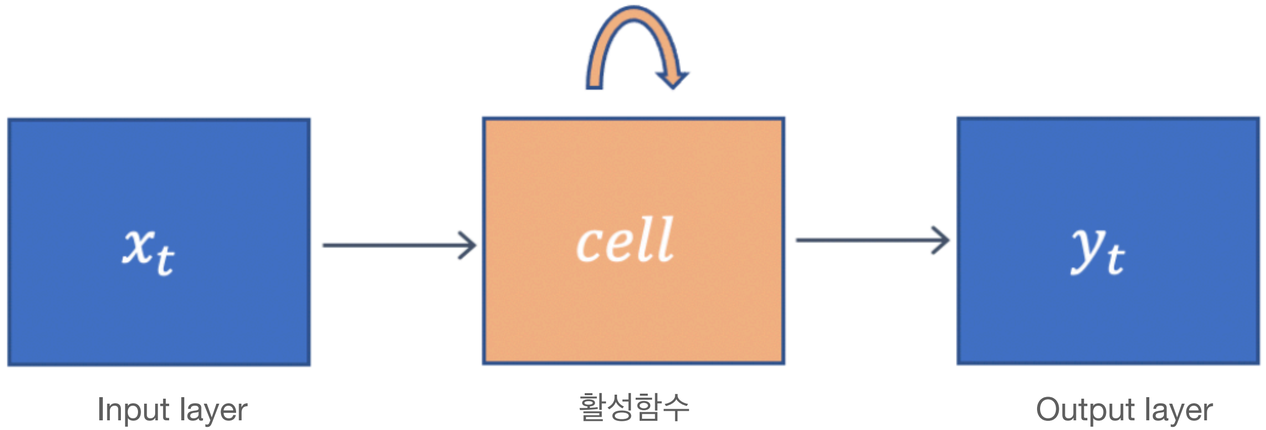
RNN (Recurrent Neural Networks)

- (텍스트나 time series data와 같은) sequential data 기반

2. RNN

Recursively Neural Network 구조

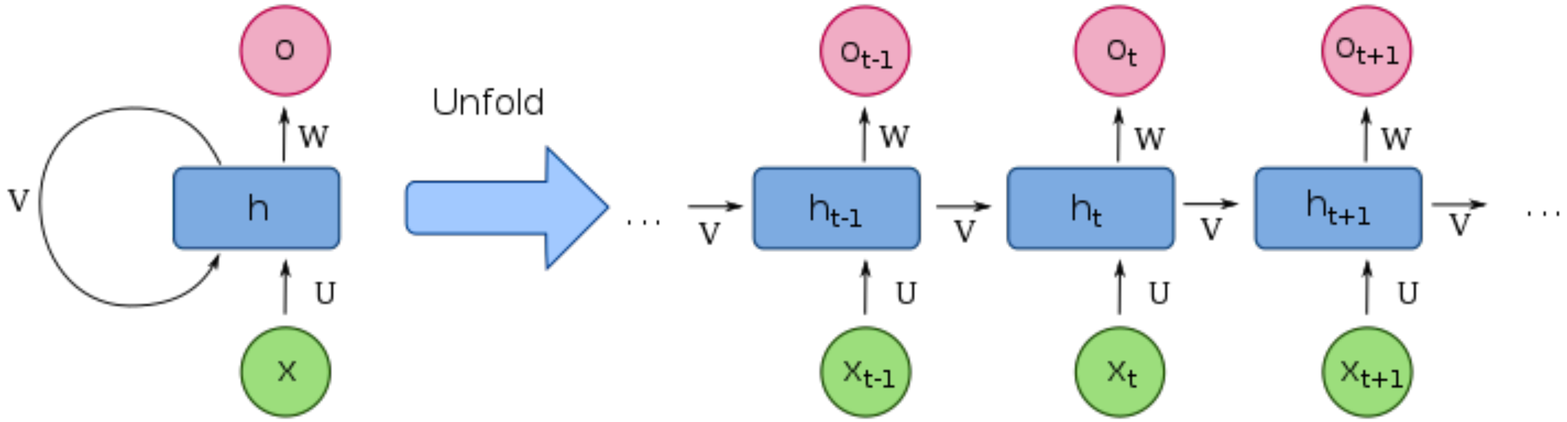
- RNN은 Hidden Layer의 노드에서 활성화 함수를 거쳐 나온 결과값을 Output Layer로 보내면서 다시 다음 Hidden Layer 노드 계산의 입력값으로 보내는 신경망
- 결과값이 다음 hidden layer 노드의 입력값 계산에 보내지는 것을 '순환한다'라고 함



2. RNN

Recursively Neural Network 구조

- Cell은 이전 time step에서의 출력값을 기억하는 역할을 수행하므로 메모리 셀 또는 RNN Cell
- Cell이 값을 기억한다는 것은 이전 time step에서 Hidden Layer의 메모리 셀의 출력값을 자신의 입력값으로써 재귀적으로(recursively) 사용



3. RNN 실습

주가예측 모델 신경망

- 입력과 출력을 시퀀스 단위로 처리

  - 시퀀스란 문장 같은 단어가 나열된 것
  - 이러한 시퀀스들을 처리하기 위해 고안된 모델을 시퀀스 모델
  - 그중에서 RNN은 딥 러닝의 가장 기본적인 시퀀스 모델

- 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할을 하는 노드를 셀

  - 이 셀은 이전의 값을 기억, 일종의 메모리 역할을 수행(**메모리 셀)**
  - 은닉층의 메모리 셀에서 나온 값이 다음 은닉층의 메모리 셀에 입력 ➔이 값을 은닉 상태

3. RNN 실습

- **금융 데이터 “ 거래소별 전체 종목 코드와 가격데이터”**

  - Pandas-datareader : 시계열 데이터 수집 라이브러리
    - 금융 데이터를 다루는데 가장 기본이 되는 데이터는 거래소별 전체 종목 코드와 가격 데이터
    -pandas-datareader는 잘 구성된 시계열 데이터 수집 라이브러리로 사용이 간편하고 다양한 시계열 데이터를 수집할 수 있다는 장점
    -거래소별(KRX, NASDAQ, NYSE 등)
    -전체 종목 코드(ticker symbol)를 가져오는 기능이 없으며, 야후 파이낸스가 더 이상지원되지 않고(deprecated), 구글 파이낸스는 UNSTABLE_WARNING + RemoteDataError
    -FinanceDataReader는 pandas-datareader 를 대체하기 보다 보완하기 위한 목적

3. RNN 실습

금융 데이터 “ 거래소별 전체 종목 코드와 가격데이터”

Pandas-datareader 주요한 기능

- 종목 코드
  - 거래소별 전체 종목코드:
  -KRX (KOSPI, KODAQ, KONEX), NASDAQ, NYSE, AMEX, S&P 500
- 가격 데이터
  -해외주식 가격 데이터: AAPL(애플), AMZN(아마존), GOOG(구글) 등
  -국내주식 가격 데이터: 005930(삼성전자), 091990(셀트리온헬스케어) 등
  -각종 지수: KS11(코스피지수), KQ11(코스닥지수), DJI(다우지수), IXIC(나스닥 지수), US500(S&P 5000)
  -환율 데이터: USD/KRX (원달러 환율), USD/EUR(달러당 유로화 환율), CNY/KRW: 위엔화 원화 환율
  -암호화폐 가격: BTC/USD (비트코인 달러 가격, Bitfinex), BTC/KRW (비트코인 원화 가격, 빗썸)
-

3. RNN 실습

- **주가예측 모델 신경망**

주식을 잘 알고 있는가?

	D-10	D-9	D-8	...	D-3	D-2	D-1
거래일	2020-03-11	2020-03-12	2020-03-13	...	2020-03-20	2020-03-23	2020-03-24
고가	88500	84600	84700	...	76200	71600	78800
저가	85100	79600	76000	...	70100	68000	72100
시가	88200	83800	76900	...	71800	69700	73000
종가	85500	82800	82500	...	74800	69400	78700
...	...	...	...	...	...	...	...
5일 이동평균선	90940.0	88520.0	86500.0	...	75580.0	73340.0	73000.0
20일 이동평균선	96535.0	95575.0	94475.0	...	87495.0	85990.0	85025.0
60일 이동평균선	96283.333333	96273.333333	96183.333333	...	94765.0	94345.0	94093.333333
120일 이동평균선	88926.666667	88955.0	88974.166667	...	88681.666667	88565.833333	88543.333333
MACD	-2358.031666	-2960.471783	-3422.663079	...	-6116.420875	-6613.634976	-6185.940782

➡

D-0 종가
5% 상승
여부 예측

1 or 0

실습

- 파이썬 금융 데이터 : Finance-data reader
  - !pip install - ㅕ finance-datareader
  - import FinanceDataReader as fdr
  - fdr.__version__

  - https://inhovation97.tistory.com/54

3. RNN 실습

전체 종목 데이터를 얻어오기 위해 거래소 심볼 확인

한국

심볼	거래소
KRX	KRX 종목 전체
KOSPI	KOSPI 종목
KOSDAQ	KOSDAQ 종목
KONEX	KONEX 종목

미국

심볼	거래소
NASDAQ	나스닥 종목
NYSE	뉴욕 증권거래소 종목
AMEX	AMEX 종목
SP500	S&P 500 종목

※ KRX는 KOSPI,KOSDAQ,KONEX 모두 포함

https://financedata.github.io/posts/finance-data-reader-users-guide.html
https://inhovation97.tistory.com/61

3. RNN 실습

- 영화 리뷰 감정 분석

3. RNN 실습

- **스팸 메일 분류: 이진 분류 문제**
  - 마지막 출력층: 1개 뉴런
  - 활성함수 : 시그모이드 함수
  - alidation_split=0.2 (훈련 데이터의 20%)
  - 교차 검증을 진행

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, None, 32)          128416

 simple_rnn_2 (SimpleRNN)    (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 130,529
Trainable params: 130,529
Non-trainable params: 0
_________________________________________________________________
```

```
model_1 = Sequential()
model_1.add(Embedding(vocab_size, 32))
model_1.add(SimpleRNN(32))
model_1.add(Dense(1, activation='sigmoid'))
model_1.summary()
```

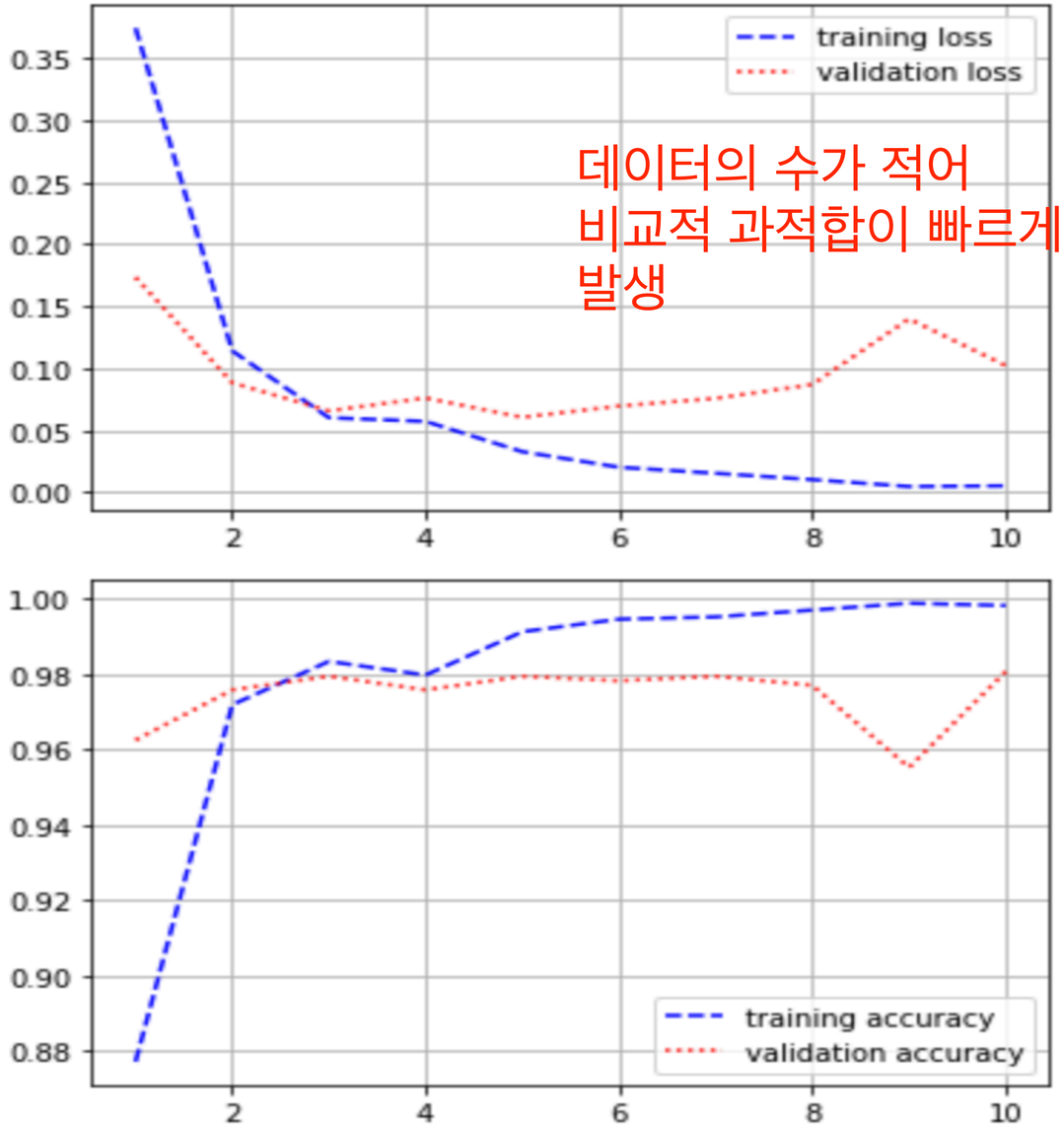
1. 스팸 메일 분류

- **스팸 메일 분류: 이진 분류 문제**
  - 마지막 출력층: 1개 뉴런
  - 활성함수 : 시그모이드 함수
  - alidation_split=0.2 (훈련 데이터의 20%)
  - 교차 검증을 진행

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_2 (Embedding)      (None, None, 32)          128416

 simple_rnn_2 (SimpleRNN)     (None, 32)                2080

 dense_2 (Dense)              (None, 1)                 33

=================================================================
Total params: 130,529
Trainable params: 130,529
Non-trainable params: 0
_________________________________________________________________
```

```
model_1 = Sequential()
model_1.add(Embedding(vocab_size, 32))
model_1.add(SimpleRNN(32))
model_1.add(Dense(1, activation='sigmoid'))
model_1.summary()
```

스팸 메일 분류



가장 적절한 epoch는 얼마인가?

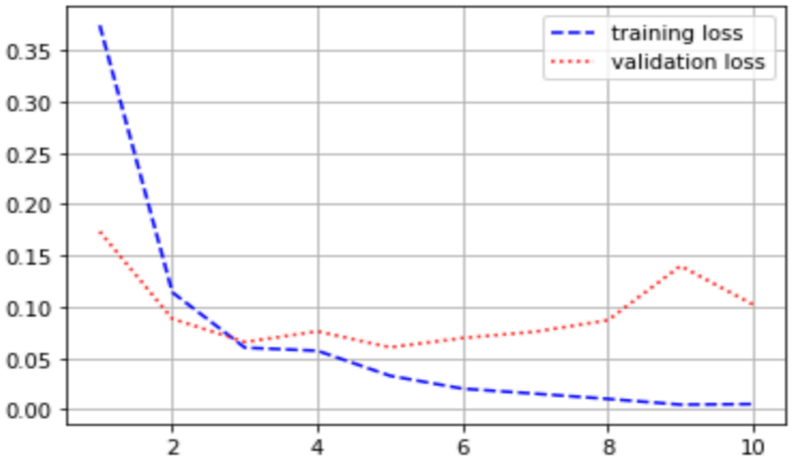
- 검증 데이터의 loss가 증가하기 시작하는 부분인 3이 제일 적절하다고 판단 됨(불균형한 데이터셋임을 확인)
- Stratified 방식을 통해 훈련 데이터와 테스트 데이터를 분리
- 분류하고 토큰화를 진행할 경우 단어에 대한 index가 같지 않을 수 있어 모델의 성능이 저하될 수 있음
- 훈련 데이터와 테스트 데이터의 타겟 변수의 라벨 분포가 비슷하게 분리됨

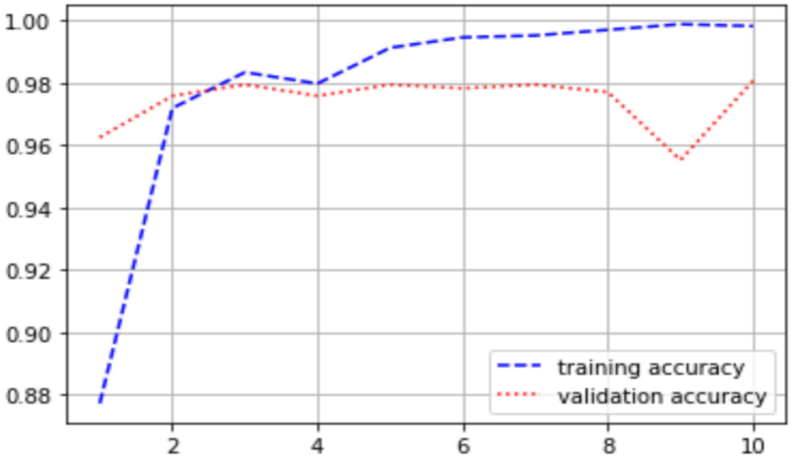
스팸 메일 분류

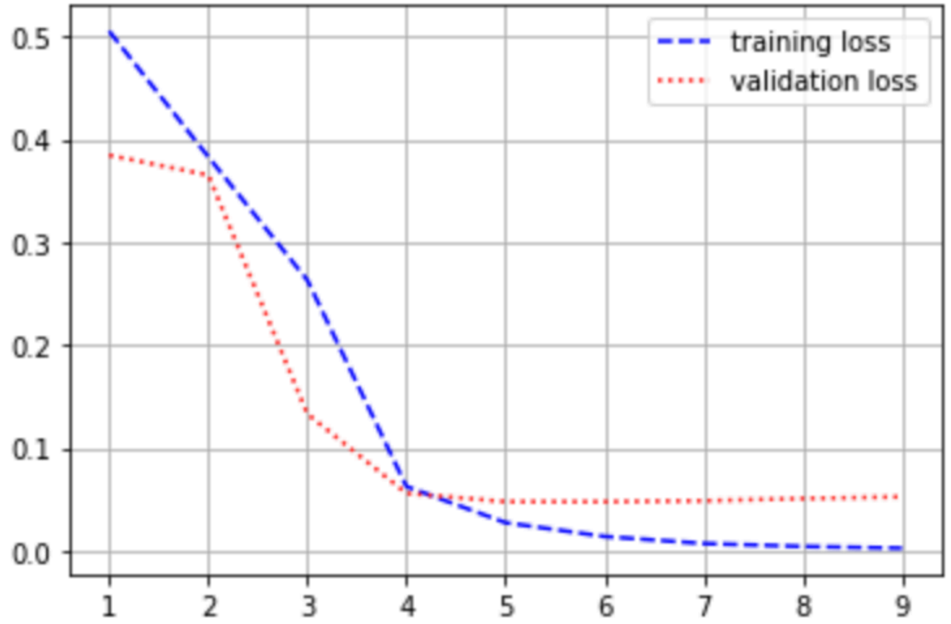
```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_11 (Embedding)     (None, None, 32)          128416

 dropout_8 (Dropout)          (None, None, 32)          0

 conv1d_4 (Conv1D)            (None, None, 32)          5152

 global_max_pooling1d_4 (Glo   (None, 32)               0
 balMaxPooling1D)

 dense_15 (Dense)             (None, 64)                2112

 dropout_9 (Dropout)          (None, 64)                0

 dense_16 (Dense)             (None, 1)                 65

=================================================================
Total params: 135,745
Trainable params: 135,745
Non-trainable params: 0
_________________________________________________________________
```

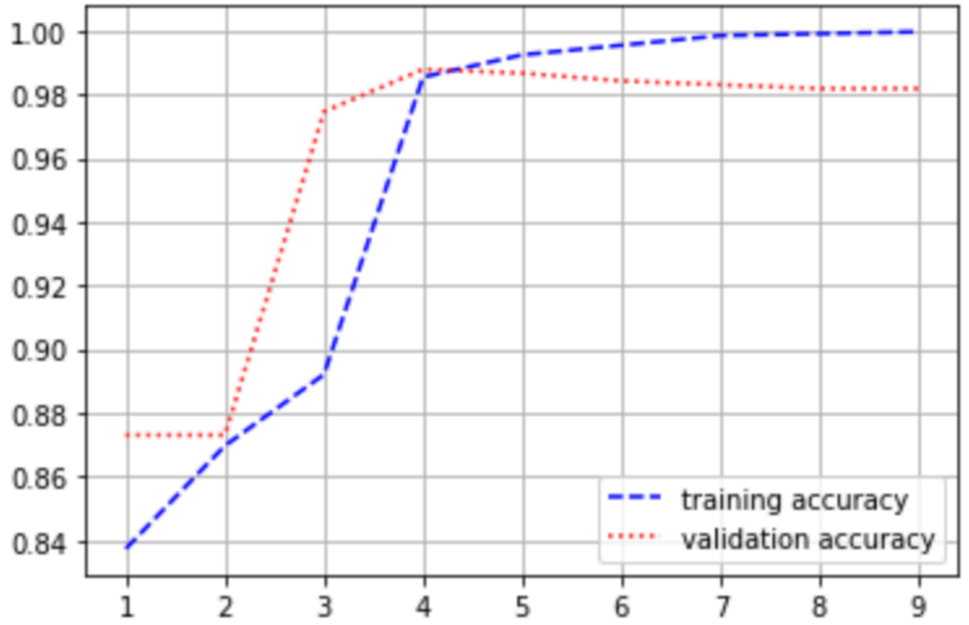
```
model_2 = Sequential()
model_2.add(Embedding(vocab_size, 32))
model_2.add(Dropout(0.2))
model_2.add(Conv1D(32, 5, strides=1,
                   padding='valid',
                   activation='relu'))
model_2.add(GlobalMaxPooling1D())
model_2.add(Dense(64, activation='relu'))
model_2.add(Dropout(0.2))
model_2.add(Dense(1, activation='sigmoid'))
model_2.summary()
```

스팸 메일 분류


training loss
validation loss
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
2
4
6
8
10


training accuracy
validation accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
2
4
6
8
10


training loss
validation loss
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6
7
8
9


training accuracy
validation accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
1
2
3
4
5
6
7
8
9

2. 감정 분석

- **스팸 메일 분류: 이진 분류 문제**
  - 마지막 출력층: 1개 뉴런
  - 활성함수 : 시그모이드 함수
  - alidation_split=0.2 (훈련 데이터의 20%)
  - 교차 검증을 진행