Keras MNIST CNN 모델링

**1. 진행 순서**

**[ GPU 사용 가능 상태 확인 및 라이브러리 import ]**



**[ 데이터셋 불러오기 및 데이터 형태 변형 ]**

텍스트이(가) 표시된 사진

자동 생성된 설명

**[ 모델링 ]**

텍스트, 영수증, 서류이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

* 파라미터 수는 약 80만개, 학습이 필요없는 파라미터 수는 350개로 정의됨.

**[ 학습과정 설정(compile) ]**

텍스트이(가) 표시된 사진

자동 생성된 설명

* Tensorflow 라이브러리를 파라미터\* 사용을 위해 불러옴.
* Loss : categorical\_crossentropy, optimizer = adam\*, metrics=[‘accuracy’]

## loss와 metrics를 실습시간에 ‘binary\_crossentropy’, ‘binary\_accuracy’로 두었더니 정확도가 급격히 상승하는 현상이 발생했었다. 하지만 이 문제는 multiclass classification 문제로 binary는 적합하지 않기 때문에 이를 수정하여 학습하도록 했다.

**[ 모델 학습 ]**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

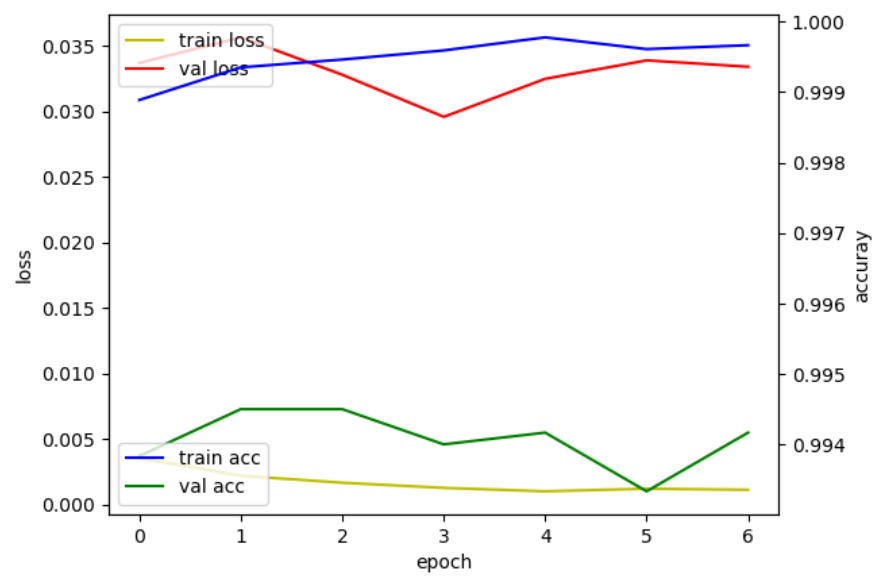
자동 생성된 설명

* 모델 학습을 통해 train 모델의 성능이 최대 99.7%, valid 모델의 성능이 최대 99.45%인 모델을 생성했다.

**[ 모델 평가 ]**

텍스트이(가) 표시된 사진

자동 생성된 설명



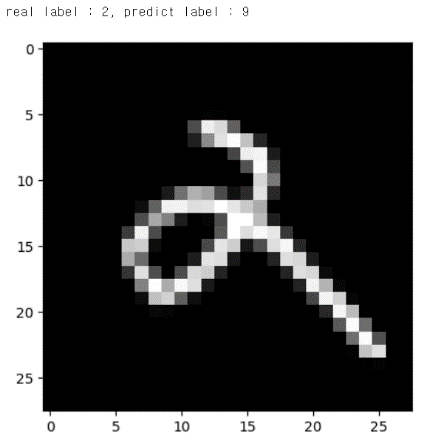
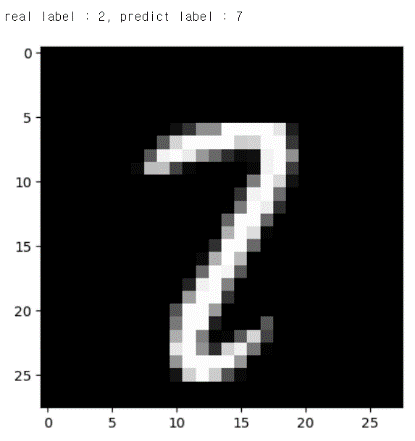
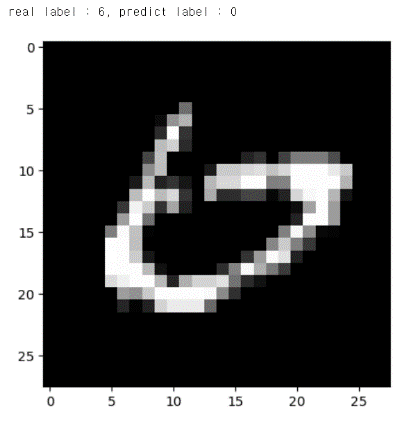
* History 확인
* epoch = 0 부터 train 모델과 test 모델의 정확도가 99%대로 들어섰다.
* 과소적합(underfitting)과 과대적합(overfitting)의 양상은 보이지 않는다.
* epoch = 1일 때 일반화 성능이 가장 좋은 것으로 보인다.

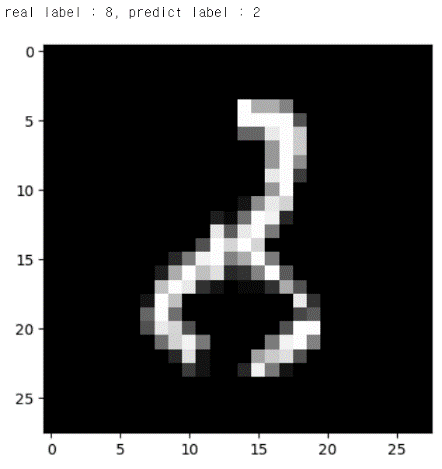
**[ 결과 분석 ]**

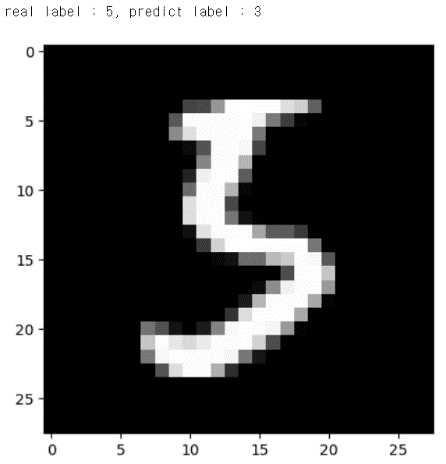
1. **잘못 분류된 케이스**

텍스트이(가) 표시된 사진

자동 생성된 설명







* 총 다섯가지의 케이스를 확인했다.
* 기존의 Y\_test[i]였던 부분을 list(Y\_test[i]).index(1)으로 변경하여 프로그래밍하였다.
* 잘못 읽을 가능성이 높은 숫자에서만 에러가 발생했음을 알 수 있었다.

1. **현실적인 최대 정확도 제안**

테이블이(가) 표시된 사진

자동 생성된 설명

* 다음의 confusion matrix를 보면 오분류된 케이스는 9994개의 데이터 중 63개이며 모델의 성능은 99.36%이다.
* 오분류된 데이터를 확인했을 때 숫자의 필기가 명확하지 않다는 점을 고려했을 때 99.36%의 성능은 단일모델로 구현했을 때 최선의 성능이라고 판단된다.

**2. 개선방향**

• 앙상블 기법 활용

* 앙상블 기법은 대체로 성능을 높이는 대표적인 방법 중 하나다.
* 따라서 여러 CNN 모델을 생성하여 도출된 결과를 앙상블할 것을 제안한다.
* layer 갯수 및 파라미터 조정
* 주어진 시간 및 컴퓨터 환경 여건 상, 다양한 시도를 해보지 못했다.
* 따라서 더 많은 layer를 두어 계산해볼 것을 제안한다.
* 또한 모델 학습과정 설정(compile)과 학습(fit)에 사용하는 파라미터 종류와 시도횟수를 늘려 계산해볼 것을 제안한다.

**3. 정리**

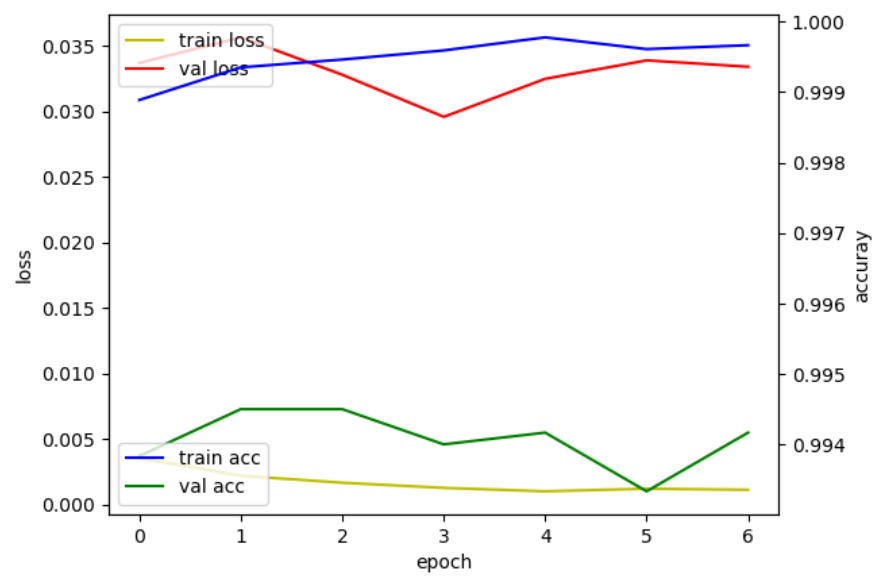
- 정확도

1) train 모델의 성능 : 최대 99.7%

2) valid 모델의 성능 : 최대 99.45%

3) test 모델의 성능 : 최대 99.31%

- 안정화



* 과소적합(underfitting)과 과대적합(overfitting)의 양상은 보이지 않는다.
* epoch = 1일 때 일반화 성능이 가장 좋은 것으로 보인다.
* Train, valid 모델 간 성능의 차이가 0.005 미만인 것으로 보아 모델은 안정적인 것으로 보인다.

**4. 진행 전략(시행 착오)**

* 하나의 모델로 바로 결정하지 않고 실습시간에 다루었던 세가지 모델과 Pytorch까지 전부 활용해보았다.
* Pytorch을 이용한 CNN 모델도 성능을 99.3%까지 끌어올렸으나 Keras를 이용한 모델에 더 시간을 많이 투자했기에 Keras 라이브러리 모델로 보고서를 작성했다.
* 구현해본 Keras 라이브러리를 활용한 CNN 모델 중 성능이 가장 좋은 Dropout 모델을 이용해 과제를 제출하기로 결정하였고, 파라미터 조정을 통해 성능을 99.4까지 올렸다.
* loss, optimizer, metrics 전부 다르게 설정하면서 성능을 비교했다. learning\_rate는 소수점자리를 낮추고 높여가면서 최적점을 찾으려 노력했다.

텍스트이(가) 표시된 사진

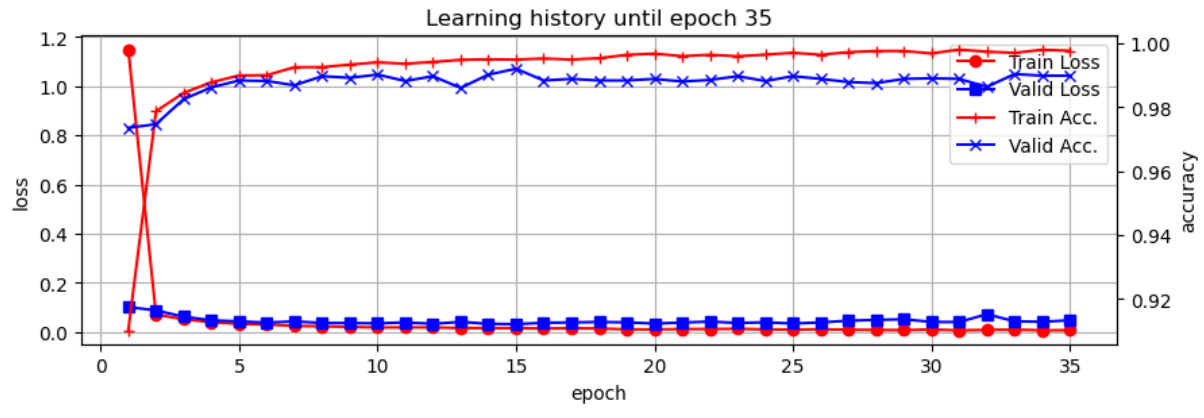
자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명