**HW3. Fashion-MNIST CRNN Classifier**

20191611 유종선

* 이번 과제는 Fashion-MNIST 데이트를 CRNN을 사용해서 classify한다. 총 10종류의 이미지를 가진 데이터 셋이다. 데이터는 28 \* 28 크기이다.
* 먼저 모델을 설명해보면, CRNN은 3개의 convolution layer와 2 fully-connected layer로 구성하였다.

Layer 1 : in = 1, out = 32

Layer 2 : in = 32, out = 64

Layer 3 : in = 64, out = 128

LSTM : embedding\_dim : 64, hidden\_dim : 128, num\_layers = 3

Fully-connected : 2304 -> 10(num\_classes)

* CRNN

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* Forward

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* Forward1, forward2 와 같이 forward 함수를 나누어 실행시켜서 단계 별로 이미지가 어떻게 변화하는지 확인해보자.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 먼저 train\_loader에 들어간 image를 2개만 뽑아보았다. 이미지는 다음과 같다.

스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명스크린샷, 직사각형, 화이트, 사각형이(가) 표시된 사진

자동 생성된 설명

1. Forward1을 실행시키고 난 후에 image 사이즈의 변화이다.







1. Forward2를 실행시키고 난 후이다.







1. Forward3 이후 부터는 이미지로 볼 수 있는 데이터 크기가 아니라는 오류가 뜬다. Image를 print해서 확인해보자.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. Forward4와 forward5는 fully-connected이므로 모두 실행시키고, 예측결과와 정답을 출력해보자.





텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

1. 이러한 식으로 image가 처리되는 것을 확인할 수 있었다. 이제 train을 해보자. Train data set은 6만개이다. Valid test를 위해서 7 : 3의 비율로 데이터를 나누어 준다. Train 42000개, valid data는 18000개로 나누어진 모습을 확인 할 수 있다.

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 데이터를 나누고, 하이퍼 파라미터는 다음과 같이 두었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. Train 실행

스크린샷, 패턴, 패브릭, 블랙이(가) 표시된 사진

자동 생성된 설명

1. Train의 accuracy와 loss, valid의 accuracy와 loss의 변화를 그래프를 보고 변화를 확인해보자. Epoch이 기준이다.

스크린샷, 라인, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

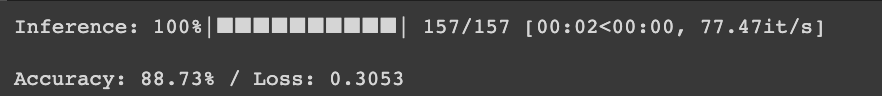
1. 마지막으로 test를 해보자. 대략 89.6% 정도의 정확도를 가지게 된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

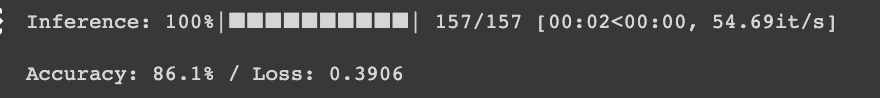
1. 정확도를 올리기 위한 과정

* Conv2d layer : 처음에는 convolution layer를 2로 두고 실행하였는데, layer를 하나 더 늘려서 모델을 구성하였다.
* Maxpooling : 현재는 3 layers 에서 maxpooling을 3번 진행한다. 그런데 현재 입력 데이터가 28 \* 28 이기 때문에, 더 적게 maxpooling을 진행해보았다. 더 많이 Maxpooling을 할 수록 정보를 너무 많이 잃을 수도 있겠다는 예상을 했다.
  + Maxpooling \*2



3번을 두번으로 줄인 결과 더 낮은 결과값이 나왔다. Maxpooling은 3번으로 유지하도록 하였다.

* Filter size : 필터 사이즈를 조절하였다. 현재 filter size는 3, 3 ,2 이렇게 진행하고 있다. Filter size가 크게 되면 이미지에서 더 큰 패턴이나 특징을 잡을 수 있는데 반해, 세부적인 정보를 놓칠 염려도 있다. 그래서 필터사이즈를 조절해서 실험을 해보았다.
  + Filter size = 2, 2 ,2



* + Filter size = 3, 3, 3



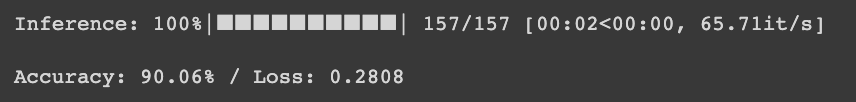
* + Filter size = 5, 5, 5

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

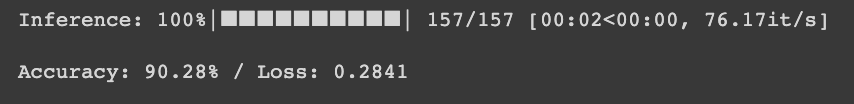
다음의 결과를 통해서, 그대로 filter\_size를 유지하였다.

* LSTM에서 hidden dimesion의 설정에 따른 결과 값을 확인해보았다. 현재는 128로 진행하였는데, 이 값을 256으로 높여서 확인해보자.
  + hidden = 256

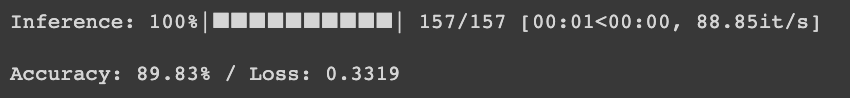


조금 더 높은 성능을 받을 수 있었다.

* Dropout : 오버피팅을 방지하고, 앙상블 메소드 효과를 주기 위해서 dropout 함수를 사용하였다. 여기서 drop out을 크기를 조절해보자
  + Drop out : 0.1



* + Drop out : 0.1, 3번의 dropout을 1번으로 줄여서 학습



Dropout 실행 결과 Drop out은 0.1로 3번 진행하는 것이 가장 좋은 결과를 얻을 수 있었다.

* hidden dimension = 256, Dropout = 0.1 로 하는 것이 성능을 높혀주었다.