**딥러닝 및 응용 assignment 보고서**

2017029516

컴퓨터 소프트웨어학부

김태환

보고서는 크게 1)모델 설명과 2)성능 향상 방법을 다루고 있다.

1. **모델 설명**

* **오토인코더 모델 코드**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

오토 인코더의 인코더는 두개의 convolution layer로 이루어져있다.

인풋 이미지가 3 \* 100 \* 100 사이즈 이므로, conv 연산의 첫번째 파라미터는 3, 그리고 채널은 32개, 커널사이즈는 3, padding은 1, stride는 1로 연산을 진행했다.

이 결과로 32 \* 100 \* 100 의 결과가 나오게 되고, 이후에 있는 pooling 연산을 통해서

32 \* 50 \* 50 이 된다.

마찬가지로 conv2 연산을 통해서 연산을 거치고 나면 16 \* 50 \* 50 의 결과가 나오게 되고, pooling layer를 거치면서 16 \* 25 \* 25 사이즈가 된다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

디코더는 convolution layer 3개와 fully connected layer 2개로 이루어져있다.

Conv 연산의 목적은 zero padding 을 통해서 사이즈를 조금 더 키워주는 역할로,

첫번째 conv layer를 거치면, 32 \* 27 \* 27 이 된다.

이후 두번째 conv layer를 거치면, 32 \* 29 \* 29 이 된다.

마지막 세번째 conv layer를 거치면, 32 \* 31 \* 31 이 된다.

이후 다시 원사이즈로 맞추어 주기 위해 flatten 연산을 진행해주고 fc layer를 거친다.

Fc layer를 두번 거쳐가며 다시 사이즈를 3\* 100 \* 100으로 만들어준다.

이후 다시 원래의 형태를 만들어주기 위해서 view 연산을 이용하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

오토 인코더는 인코더와 디코더를 연결해주며, 제공된 기본 코드와 다름이 없다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Learning rate은 0.001, 모델의 loss는 MSE를 사용했다.

* **오토 인코더 학습 코드**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

오토인코더의 학습에 epoch는 15를 주었다.

학습시간을 효율적으로 쓰기 위해 적한 값이었다.

regularization등의 기법은 사용하지 않고, 학습을 진행했다.

* **Classifier 모델 코드**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Classifier는 1개의 convolution layer와 2개의 fully connected layer로 구성했다.

Conv layer에서는 16 \* 25 \* 25 사이즈의 데이터를 32 \* 25 \* 25 사이즈로 바꾸어준다.

이후 pooling layer를 거쳐서 32 \* 5 \* 5 로 만든다.

이후 Fc layer에서 입력받을 수 있도록 flatten 연산을 거쳐서 데이터를 펴준다.

이후의 fc layer 두개를 거치며 최종적으로 102개로 분류하는 분류기가 완성된다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

오토인코더와 마찬가지로 lr 은 0.001을 썼다. 그리고 분류 문제이므로 loss는 CrossEntropy를 썼다.

* **Classifier 학습 코드 및 정확도 측정**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Epoch은 30으로 진행했다.

그리고 classifier가 학습할때는 인코더는 학습을 하면 안되므로 , model.eval을 호출하여 오토인코더의 상태를 학습에서 평가로 바꾼다.

이후 model2(classifier)를 학습상태로 바꾸고 학습을 진행한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

정확도 측정에서는 model2(classifier)의 상태를 학습에서 평가로 바꾸고, 정확도 측정을 진행한다.

* **정확도**



정확도는 20 % 가 나왔다.

1. **성능 향상 시도**
2. **Batch norm**

성능향상을 위해 가장 먼저 시도한 것은 batch norm 기법이었다.

Batch norm은 Classifier의 linear network에 추가하였다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Classifier에서 flatten 작업이후 batch norm을 적용하여, 첫번째 linear network로 들어가는 인풋을 normalization하는 방법으로 구현하였다.

이렇게 기법을 적용한 후 학습을 시켜보니, 정확도가 27%로 증가했다.



1. **Drop out**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Linear layer 사이에 dropout을 만들어서 0.5의 prob으로 drop out을 진행하고 정확도를 측정해보았다.

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

정확도는 이전과 차이가 없었다.

**2-2.** **Drop out + epoch 조절**

Epoch 수가 많거나 부족해서 차이가 없을까 싶어서 epoch을 20, 40 으로 조정하여 다시 한번 학습시켜서 정확도를 확인해 보았다.

Epoch가 20일 때,

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Epoch가 40일 때,

텍스트, 폰트, 화이트, 그래픽이(가) 표시된 사진

자동 생성된 설명

Epoch가 늘어도 성능차이는 큰 변화는 없었다.

1. **모델 아키텍처 layer 추가**

인코더 부분은 현재 conv layer 두개로 이루어져있다.

Conv layer 를 한 개 더 쌓아서 encoder 부분의 성능을 높여보고자 시도해보았다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Conv3 레이어를 추가해서 기존의 conv 연산 사이에 넣어 구현해보았다.

학습을 진행하니, 정확도는 27%로 동일하게 나왔다.



1. **디코더 batch norm 적용**

디코더의 linear layer에 batch norm을 적용하여 보았다. 디코더는 분류기에 쓰이지는 않지만 인코더와 디코더가 학습될 때 디코더의 학습이 인코더에 영향을 주므로 인코더의 성능이 더 증가하지 않을까 하는 생각에서 비롯되었다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Flatten 후 bn layer를 거치며 batch norm을 적용해보았다.

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

결과의 차이는 없었다.

1. **Learning rate 변화주기**

Learning rate 변화를 주어 더 적절한 학습을 시켜보고자 시도해보았다.

기존의 learning rate는 0.001이므로, 더 크게(0.003)와 더 작게(0.0007) 성능을 평가해보았다.

0.003인 경우엔



성능이 매우 떨어졌다. 아마도 오버피팅이 크게 작용한 것 같다.

0.0007인 경우엔



이 경우 또한 정확도가 조금 감소했다. 0.001이 optimal 한 lr에 가까운 값인 것 같다.

1. **Batch size 변화주기**

기존엔 batch size를 64로 진행했었다. 더 빠른 학습을 위해 설정했던 값인데, 128로 배치 사이즈를 늘려서 학습을 진행해보고자 한다.



결과는 차이가 없었다.