지능시스템설계 Homework 8

학번 : 2021220699
학과 : 전자전기공학부
이름 : 이은찬

과제 개요

과제

지능시스템설계 › 과제

제목	Homework 8
제출방식	온라인
게시일	2021.05.06 오전 6:53
마감일	2021.05.14 오후 11:59
배점	100
지각제출	허용 (마감일 : 2021.05.15 오전 9:59)

Chapter 8에 있는 코드들을 MNIST 와 FashionMNIST 에 적용한다.
1) train_deepnet.py 를 실행 (deep_convnet.py 가 필요함)
2) misclassified_mnist.py 실행
3) MNIST, FashionMNIST 에 대하여 반복
제출물: submit a zip file hw8.zip of python files that are necessary to execute your codes, and hw8-doc.pdf to lms.knu.ac.kr
copy policy: carbon & regular
*최종성능은 점수에 영향 거의 없음. 리포트가 중요.
점수에 대한 문의는 오혜준 조교에게 해 주기 바랍니다.
ohjm240@gmail.com

< 레포트 채점 기준 및 점수 분포 >

- 코드에 대한 설명보다는 "결과에 대한 분석을 어떻게 하였는지 / 얼마나 다양한 조합으로 실험을 수행하였는지"를 위주로 채점하였습니다.

본문

1. MNIST

1) train_deepnet.py 실행 결과

train_deepnet.py 코드는 원하는 데이터를 불러오고 이를 common에 구현되어 있는 Trainer함수를 통해 network가 높은 정확도를 가지도록 훈련시키고 훈련 종료 후 네트워크의 최적화된 가중치를 load/save 할 수 있도록 하는 코드입니다.

따라서 중요한 포인트는 Trainer의 하이퍼 파라미터를 적절히 조절하는 것입니다.

저는 기존의 batch_size를 100에서 128로 바꾸었고 epoch 수도 20에서 12로 줄여서 실행하였는데 이유는 아래와 같습니다.

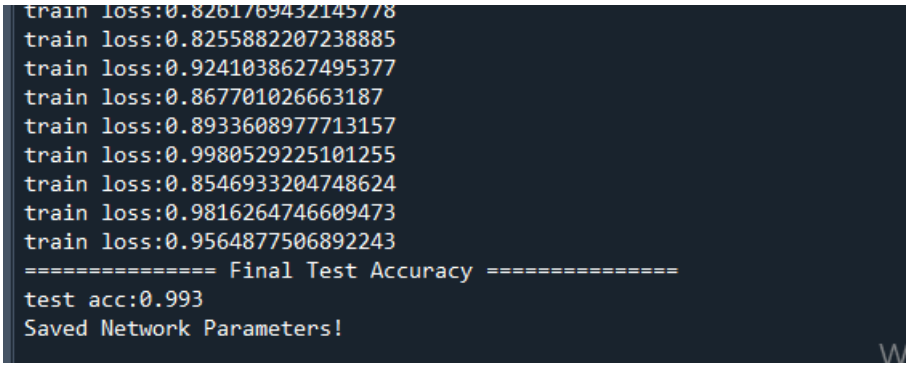
- MNIST 픽셀 너비가 28*28이라는 점에서 Batch_size를 128로 자주 쓴다고 합니다. 그 점에 착안해서 바꾸어 실행해 보았습니다. 여러가지 경우의 수를 둬서 해보고 싶지만 train_deepnet.py 훈련 시간이 한번에 **4시간 이상**씩 걸리는 것을 감안하면 불가능하다고 판단했습니다.
- 그로 인하여 epochs도 20에서 12로 소폭 감소시키게 되었습니다. 훈련시간을 6시간에서 4시간으로 줄일 수 있었습니다.

그리고 훈련이 끝났을 때 이 가중치를 ***deep_convnet_params.pkl***에 저장해두도록 하여 이후 2)에서 사용하고자 하였습니다.

```
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False)

network = DeepConvNet()
trainer = Trainer(network, x_train, t_train, x_test, t_test,
                  epochs=12, mini_batch_size=128,
                  optimizer='Adam', optimizer_param={'lr':0.001},
                  evaluate_sample_num_per_epoch=1000)
trainer.train()

# 매개변수 보관
network.save_params("deep_convnet_params.pkl")
print("Saved Network Parameters!")
```

마지막으로 Trainer 코드를 통해 출력되는 Test accuracy는 99.3%가 나왔습니다.

하이퍼 파라미터 조합이 매우 좋음을 알 수 있습니다.

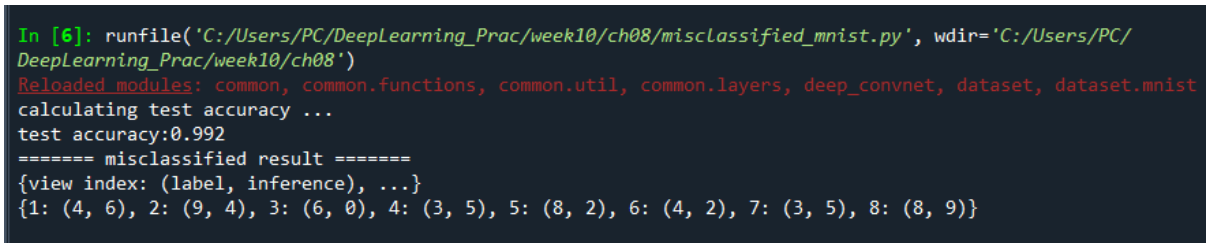


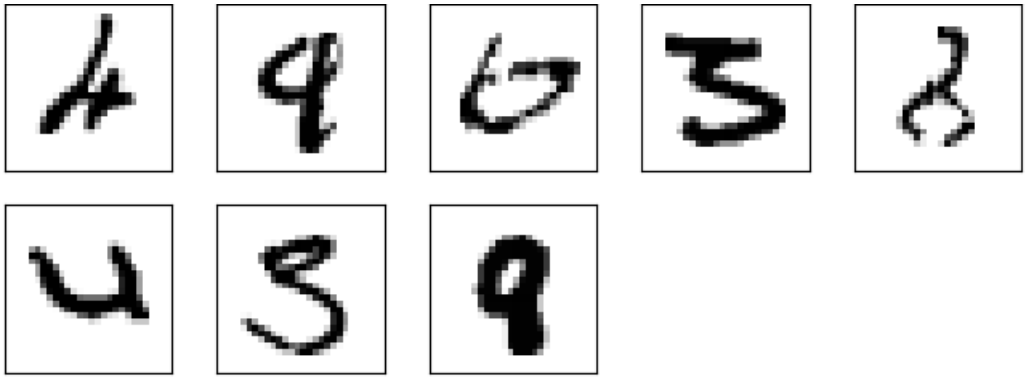
2) misclassified_mnist.py 실행 결과

이번 코드는 앞서 훈련된 pkl 가중치 모델을 불러와서 1000개의 sample data에서 misclassified 된 데이터들을 찾아내고 misclassified된 데이터의 값이 어떤 값이었는지 출력해주는 예제 코드입니다.

결과는 아래와 같았습니다.

- Test Accuracy는 99.2%
- 출력되는 데이터의 이미지는 충분히 혼동될만한 데이터 이미지임을 직접 눈으로 확인할 수 있었으며, 출력되는 misclassified value들 역시 매우 타당한 misvalue임을 육안으로 직접 확인이 가능했습니다.





2. Fashion MNIST

0) Fashion MNIST 전처리하기

사이킷런(sklearn) 패키지를 임포트하여 fashion_mnist 데이터 셋을 불러오는 방법을 사용하였습니다.

load_data()를 통해 데이터를 불러 온 이후에는 사용할 Network 모델(cnn)이 MNIST 데이터 크기에 맞춰져 있는 것을 생각했습니다.

- n개의 MNIST set: (n, 1, 28, 28)
- n개의 F-MNIST set: (n, 28, 28)

데이터 크기는 위와 같았습니다.

이 때문에 모델을 수정하는 것과 F-MNIST의 크기를 기존과 동일하게 Reshaping해주는 방법이 있었고 저는 F-MNIST의 크기를 Reshaping해주는 방식을 적용하여 Network 모델을 바꾸지 않고 사용할 수 있었습니다.

넘파이의 np.newaxis 함수를 사용하여 행의 차원을 1 늘린 다음,

reshape 함수를 통해 Fashion-MNIST 데이터의 SIZE를 (n, 28, 28) → (n, 1, 28, 28)으로 맞추어 주었습니다.

이후 fixel value들을 Normalization 시켜 주었습니다. (x를 255로 나누어줌)

```
from sklearn.datasets import fetch_openml
#Fashion MNIST를 불러온다.
fashion_mnist = keras.datasets.fashion_mnist
(x_train, t_train), (x_test, t_test) = fashion_mnist.load_data()


#입력 데이터 전처리
#차원 추가를 통해 3D -> 4D & reshape 과정 (60000,28,28) -> (60000,1 ,28,28)
x_train= x_train[np.newaxis]
x_test= x_test[np.newaxis]

x_train = x_train.reshape(len(x_train[0]),1,28,28)
x_test = x_test.reshape(len(x_test[0]),1,28,28)



x_train = x_train/255
x_test = x_test/255
#Train Fashion MNIST!
network = DeepConvNet()
```

1) train_deepnet.py 실행 결과

MNIST case와 같은 이유로 하이퍼파라미터를 batch_size=128, epochs = 12로 수정하였습니다.

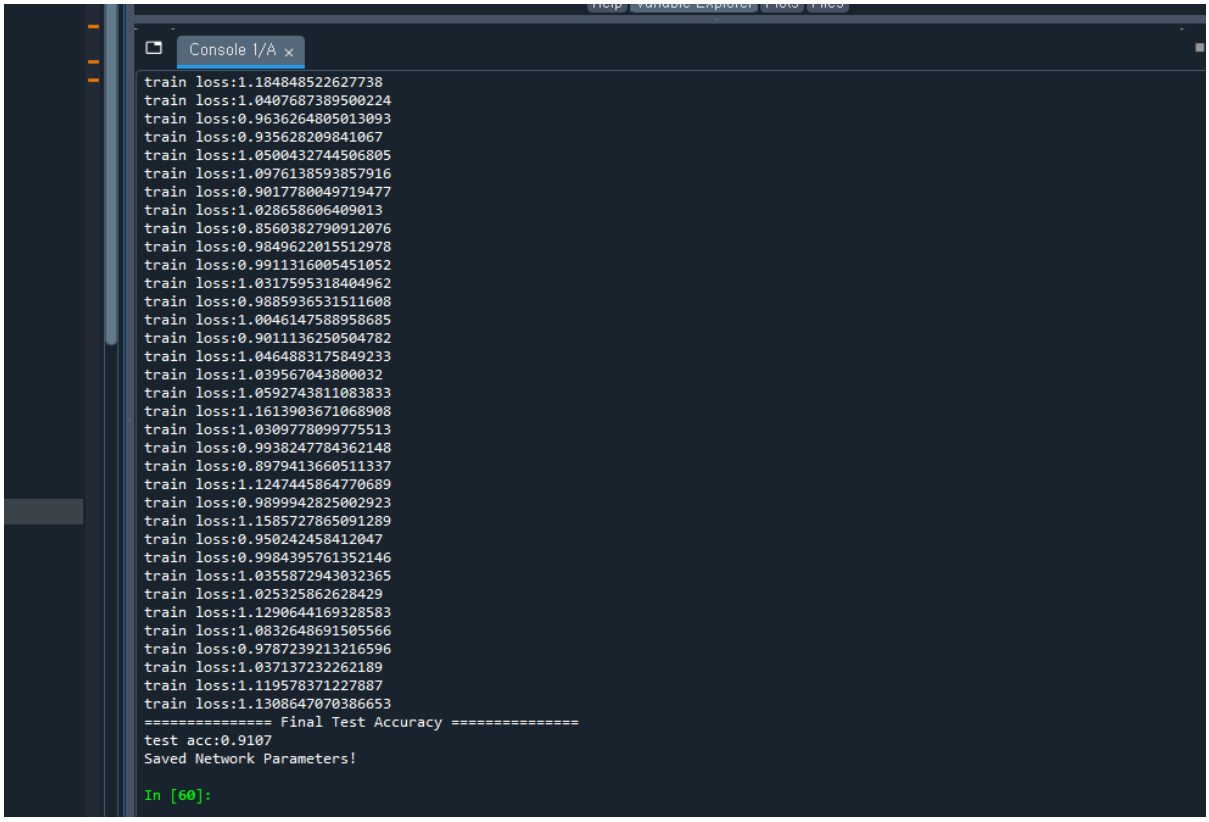
그리고 훈련 종료 시 가중치 값들을 ***fMNIST_deep_convnet_params.pkl***에 저장해 두도록 하였습니다.

```python
## 가중치를 조절하면서 성능 뽑아내기
# ex) epochs = 12 , batch_size =128 ...

trainer = Trainer(network, x_train, t_train, x_test, t_test,
                  epochs=12, mini_batch_size=128,
                  optimizer='Adam', optimizer_param={'lr':0.001},
                  evaluate_sample_num_per_epoch=1000)
trainer.train()

# 매개변수 보관
network.save_params("fMNIST_deep_convnet_params.pkl")
print("Saved Network Parameters!")
```

훈련시간은 대략 4~5시간 정도가 소요되었으며

Trainer()를 통해서 출력되는 Test Accuracy는 91.07%였습니다.

- MNIST Data에 비해서는 정확도가 낮았습니다.
- 일반적으로 Google 검색을 통해 외부 자료를 찾아봤을때 FashionMNIST가 MNIST에 비해 낮은 정확도가 나오는 것은 당연하다고 합니다.
- Network 모델 구조을 바꾸지 않는 이상은 성능을 비약적으로 올리기는 불가능하다고 판단했고 더이상 훈련을 진행하지 않았습니다.



2) misclassified_mnist.py 실행 결과

코드 설명은 1.MNIST의 2)와 동일하므로 생략하였습니다.

결과는 아래와 같았습니다.

MNIST에 비해 Fashion-MNIST 데이터가 복잡함을 여기서 알 수가 있습니다.

MNIST는 0,1,2,3,4,5 ... 등등 사실 형체적으로 매우 비슷한 숫자는 없는 반면

아래와 같이 Fashion-MNIST는 신발 관련 데이터 타입이 3종류 상의 관련 데이터 타입이 4종류 있습니다.

- 신발 관련 : 5,7,9
- 상의 관련 : 2,4,6,0

```
Fashion MNIST label

0 ~ 9 exists

0 = T-shirt/top 1 = Trouser
2 = Pullover    3 = Dress
4 = Coat        5 = Sandal
6 = Shirt       7 = Sneaker
8 = Bag         9 = Ankle boot
'''
```

이 때문에 아래 misclassified된 결과는 아래와 같았습니다.

출력을 보면, 전반적으로 신발 데이터는 다른 신발에 대해 misclassified가 발생되었고

상의 데이터는 다른 상의 관련 데이터로 misclassified되는 점을 확인할 수 있었습니다.

Test Accuracy는 91.07%로 위에서 설명 드린 MNIST→ F-MNIST에서 증가되는 복잡도를 생각했을 때는 매우 높은 정확도 수치라고 볼 수 있을 것입니다.

```
IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PC/DeepLearning_Prac/week10/ch08/james_test.py', wdir='C:/Users/PC/DeepLearning_Prac/week10/ch08')

2021-05-11 18:28:10.373213: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library
'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-05-11 18:28:10.373428: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a
GPU set up on your machine.
calculating test accuracy ...
test accuracy:0.9107
======= misclassified result =======
{view index: (label, inference), ...}
{1: (7, 5), 2: (4, 2), 3: (9, 5), 4: (4, 2), 5: (3, 6), 6: (2, 6), 7: (4, 6), 8: (4, 6), 9: (4, 6), 10: (2, 3), 11: (9, 7), 12: (4,
2), 13: (9, 7), 14: (4, 2), 15: (0, 6), 16: (6, 3), 17: (4, 3), 18: (4, 3), 19: (4, 2), 20: (4, 6)}
```

3. Summary

이번 예제와 과제를 통해서 아래 내용들을 공부할 수 있었습니다.

- 실제로 Deep한 CNN Network의 구조와 코드를 볼 수 있었고 매우 높은 정확도 수치를 얻을 수 있었습니다.
- Parameter를 저장해두어 훈련을 여러번 하지 않고 유용하게 사용하는 방법
- MNIST와 달리 Fashion MNIST의 정확도가 99를 찍지 못하는 어느정도의 고찰을 해볼 수 있었습니다.