과제 04 - validation set

뉴럴 네트워크 학습 구현 과제

학번: 20195234

이름: 임지운

소속: 빅데이터 전공

- 문제 1. 초기 학습 데이터를 train set과 validation set으로 분할

```
# 데이터 가져오기
##############################################################################
(train_data_X, train_data_Y), v, (tx, ty) = mnist_loader.load_data('./data/mnist.pkl.gz')
train_data_Y = one_hot(train_data_Y, size=10)
ty = one_hot(ty, size=10)
train_data_X = np.reshape(train_data_X, [-1, 28, 28, 1])
tx = np.reshape(tx, [-1, 28, 28, 1])

print("data_X : ", train_data_X.shape)
print("data_Y : ", train_data_Y.shape)

# 데이터 분할하기
train_data_X, val_data_X, train_data_Y, val_data_Y = train_test_split(
    train_data_X, train_data_Y, test_size=0.2, random_state=1234)

print("train_data_X : ", train_data_X.shape)
print("train_data_Y : ", train_data_Y.shape)
print("val_data_X : ", val_data_X.shape)
print("val_data_Y : ", val_data_Y.shape)
```

기존에 mnist_loader.load_data로 가져온 데이터셋을 훈련 데이터와 태스트 데이터로 분할하는 과정이 있다. 여기서 훈련 데이터(train_data_X, train_data_Y)를 sklearn.model.selection의 train_test_split을 사용해 분할한다. 데이터의 80%는 train set(훈련데이터)로 사용하고 나머지 20%는 validation set(검증데이터)으로 사용한다. 여기서 random_state는 난수 생성 seed로 매번 학습 데이터를 분할할 때 같은 결과로 분할하기 위해 사용한다.

아래는 해당 코드의 결과이다.

```
C:\Users\kjhgf\anaconda3\python.exe
Connected to pydev debugger (build
data_X :  (50000, 28, 28, 1)
data_Y :  (50000, 10)
train_data_X :  (40000, 28, 28, 1)
train_data_Y :  (40000, 10)
val_data_X :  (10000, 28, 28, 1)
val_data_Y :  (10000, 10)
```

기존 train set 50000개 중 80%를 새로운 train set으로 넣어주어 40000개가 들어갔고 나머지 10000개의 데이터는 validation set에 들어간 것을 확인할 수 있다.

- 문제 2. Early stopping 구현

  학습 시 매 epoch마다 전체 train set과 validation set에 대한 정확도 평가를 하여 일정 epoch, 학습 횟수마다 validation 정확도가 더 이상 최대치를 갱신하지 못하면 반복을 정지시켜 모델이 train set에 대하여 과적합을 막는 과정이다.

  현제 내 모델은 total_iter이 1000이고 10회 학습 시 중간에 train_acc와 val_acc를 계산하여 출력하고 있다. 나는 val_acc가 계산될 때 연속해서 3번 갱신이 되지 않는다면 학습을 중단시키는 Early stopping을 구현하였다.

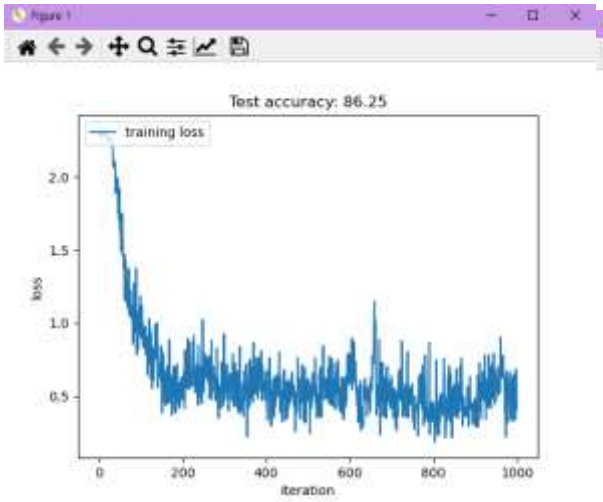
```
if iter % 10 == 0:
    train_acc = accuracy(net, batch_train_X, batch_train_Y)
    val_acc = accuracy(net, val_data_X, val_data_Y)
    train_accuracy_history.append(train_acc)
    val_accuracy_history.append(val_acc)
    print(
        f'Iteration: {iter}, Loss: {loss[-1]:.6f}, Training
    if val_acc < max(val_accuracy_history):
        count += 1
        if count == 3:
            total_iter = (iter+10)
            break
    else:
        count = 0
```

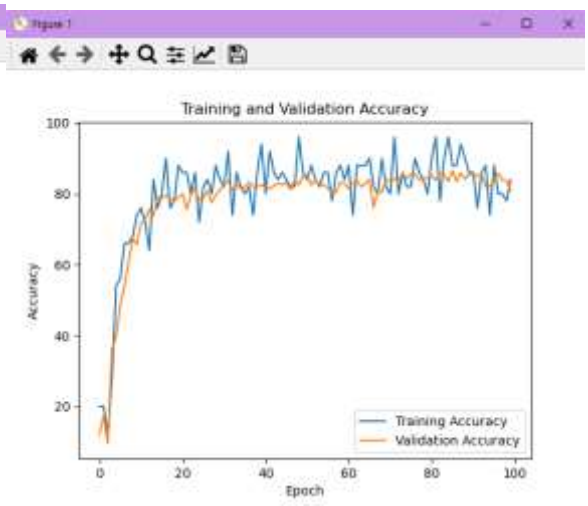
  간단하게 변수 하나와 if문으로 구현하였다.

  해당 코드의 동작은 문제 3번에 첨부할 예정이다.

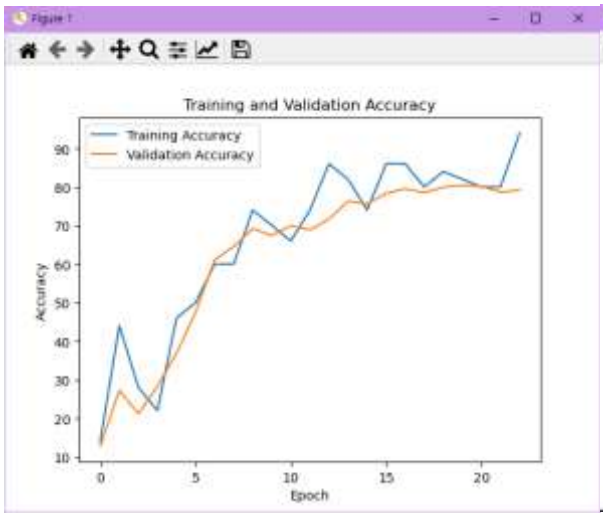
- 문제 3. Early stopping 적용 전후 사례를 보이시오.

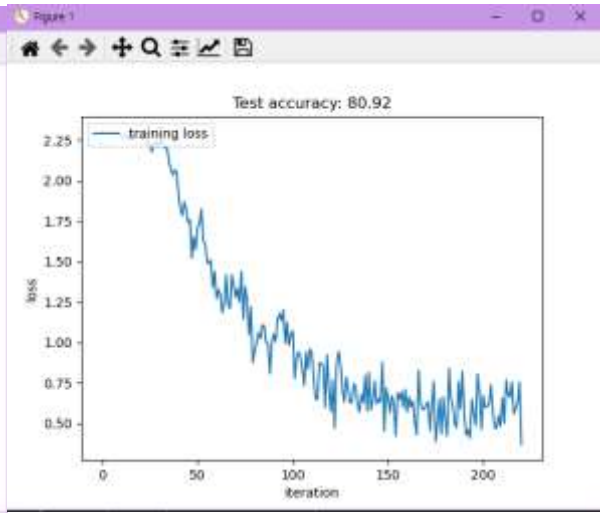
  아래 그림은 early stopping을 적용하기 전의 결과를 캡쳐한 것이다.





  아래 사진은 early stopping을 적용한 후의 결과이다.





- 문제 4. 실험 결과에 대한 고찰

  Early stopping이 필요한 가장 큰 이유는 과적합을 막기 위해서이다. 학습을 하면 할 수록 train set에 대한 정확도가 올라가지만 validation set에 대한 정확도는 점점 낮아진다. 이를 막기 위해 validation set에 대한 정확도를 모니터링하고 더 이상 validation set의 성능이 개선되지 않는다면 학습을 조기 종료하면서 과적합을 방지하고 최적의 파라미터를 얻는다.

또한 시간과 자원을 절약할 수 있다. 보통 해당 학습의 10epoch당 걸리는 시간은 대략 3분 정도 걸렸다. 이를 1000epoch를 학습시키면 300분, 5시간이나 걸린다는 의미이다. 하지만 조기 종료는 대략 200~300epoch 사이에서 일어나므로 시간을 약 210분~240분 가량을 아낄 수 있고 해당 시간만큼의 컴퓨터 자원도 아낄 수 있다.

마지막으로 일반화된 모델을 얻을 수 있다. 과적합이 발생한 모델로 새로운 데이터나 다른 데이터에서 사용할 경우 낮은 정확도를 보일 수 있지만 조기 종료로 일반화된 모델을 사용하면 좋은 결과를 얻을 수 있다.

보통 위 3가지 이유가 early stopping을 사용하는 이유이다.

하지만 내 경우 early stopping이 하이퍼 파라미터를 얻는데 큰 효과를 보지 못했다. 모델이 과적합할 정도로 성능이 뛰어나지 않았기 때문에 학습 도중에 계속해서 일정 성능에서 진동을 했고 보통은 validation set의 평가가 낮아져야 했지만 일정하게 점수를 유지했다. 좀더 뛰어난 모델을 가지고 있었다면 이런 문제가 없었을 것 같다.