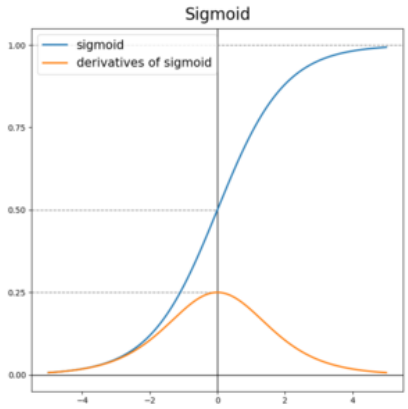
예측애널리틱스 Homework #6

고려대학교
KOREA UNIVERSITY

대학	고려대학교 공과대학
학과	산업경영공학부
학번	2017170819
이름	박상민

1. Classification 을 위한 뉴럴 네트워크 모델에서 사용하는 활성화 함수(activiation function)의 특징에 대해 조사하시오 (e.g: sigmoid, tanh, relu, leaky relu, elu, maxout 포함)

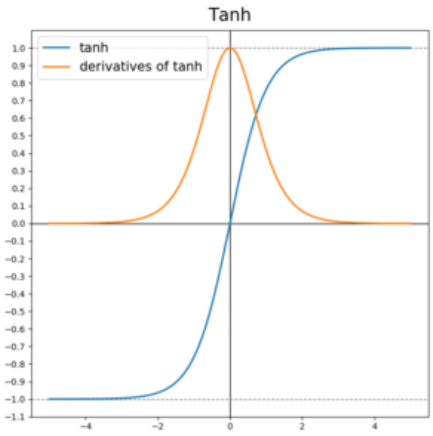
1. Sigmoid



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

sigmoid 함수는 logistic 또는 logit 함수로도 불리는 함수이다. 모든 input 값들을 0 과 1 사이의 output 값으로 변환시켜준다. 그림을 살펴보면 값이 0 에 가까워지거나 1 에 가까워지면 기울기가 아주 작아진다. 이로 인해 Vanishing Gradient 현상이 발생한다. 이 현상은 다중 layer 형성 후 활성화 함수로 sigmoid 함수를 사용할 때, 역전파 시 기울기가 거의 0 에 가까운 몇몇 노드에 의해서 입력층 쪽으로 갈수록 gradient 가 0 에 수렴하게 된다. 결국 입력층 쪽 노드들은 기울기가 사라지게 되므로 학습이 되지 않는 결과가 발생한다. 하지만 이진 분류 model 에서는 출력층 노드가 1 개이므로 사용자가 정한 threshold 에 의해 sigmoid 가 0 또는 1 의 값으로 출력할 수 있다. 따라서 이 때는 sigmoid 함수를 사용하고, 나머지 대부분의 모델에서는 사용하지 않는다.

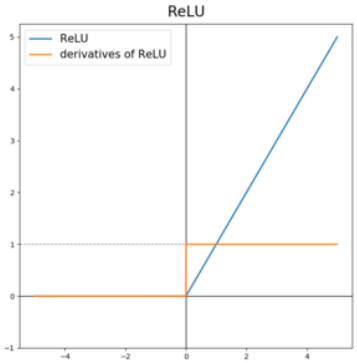
2. tanh



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

tanh함수는 모든 input 값을 -1과 1사이의 output 값으로 변환해준다. sigmoid 함수와 비슷한 형태를 가지지만 sigmoid 함수보다는 전반적으로 좋은 성능을 보여준다. 하지만 이 함수 또한 기울기가 0에 가까워지는 구간이 존재하기 때문에 sigmoid 함수에서 나타나는 Vanishing Gradient 현상이 나타난다.

3. ReLU

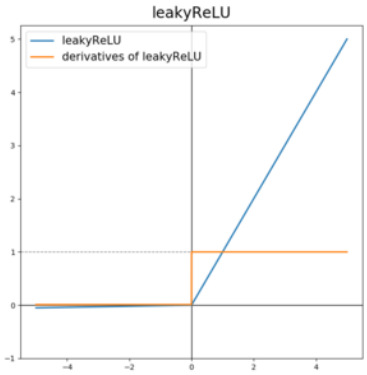


$$ReLU = \max(0, x)$$

Rectified Linear Unit의 약자로 0보다 작은 input 값에서는 0을 반환하고 0보다 큰 input 값에서는 input 값을 그대로 출력해주는 함수이다. Relu 의 도함수를 보면 0 또는 1 의 값을 가지는 것을 확인할 수 있다. 따라서 Relu 에서는 위 sigmoid와 tanh 에서 나타나는 Vanishing Gradient 현상이 발생하지 않는다. input 값이 음수일 때는 Gradient가 0이지만, 실제로 hidden layer 의 노드 값들은 대부분 0 보다 크기 때문에 이 경우는 거의 없다. 또한 함수의 형태가 간단하여, 예를 들어 지수 함수와 같은 초월 함수를 사용하지 않기

때문에 sigmoid, tanh 함수보다 약 6 배정도 빠르게 학습이 가능하다. 따라서 뉴럴 네트워크 모델에서 가장 많이 사용하는 활성화 함수 중 하나이다.

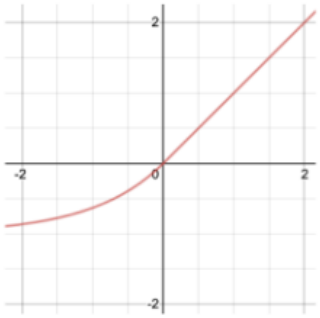
4. Leaky ReLU



$$Leaky-ReLU = \max(0.1x, x)$$

ReLU 함수와의 차이점은 input 값이 음수일 시 0 대신 값을 출력해주는 것이다. 따라서 input 값이 음수일 때도 기울기가 0 이 아닌 0.1 의 값을 가지게 된다. ReLU 보다 학습이 잘 되지만 많이 사용되지는 않는다.

5. ELU



$$ELU = \begin{cases} x & x > 0 \\ a(e^x - 1) & x \leq 0 \end{cases}$$

Exponential Linear Unit 의 약자로 ReLU 에서 파생되었으므로 역시 거의 비슷한 형태를 가진다. 지수함수를 이용하여 input 값이 음수일 경우 부드러운 곡선 형태의 output 을 보여준다. ReLU 함수의 장점을 모두 가지고 있다. 도함수가 smooth 한 형태를 가진다.

6. Maxout

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$$

Maxout 함수는 ReLU 와 Leaky ReLU 를 일반화한 함수이다. 이 함수는 두 선형 식으로 구성되어 있기 때문에 기울기 소멸 문제가 발생하지 않는다. 하지만 학습해야 하는 모수의 수가 증가하는 한계점을 가지고 있다. ReLU 의 모든 장점을 가지며, 문제점까지 해결하였다. 입력으로 들어온 값들 중 큰 값을 사용한다. 성능은 제일 좋다고 알려져 있으나, 계산 비용이 든다는 단점이 있다.

2. torchvision 패키지에 내장되어 있는 KMNIST 데이터셋을 사용하여 다음을 분석하시오. 뉴럴 네트워크 모델의 하이퍼파라미터를 명시하고 이를 변경하며 정확도 성능, 학습 상태들이 어떻게 변하는지 분석하세요.

KMNIST 데이터셋은 70000 자의 히라가나 글자 이미지와 종류(10 classes)를 기록한 데이터셋이다. 총 클래스 10 개는 각각 다음과 같다.

お	き	す	つ	な	は	ま	や	れ	を
o 오	ki 키	su 스	tsu 츠	na 나	ha 하	ma 마	ya 야	re 레	wo 오

각 글자당 7000 개의 손글씨가 저장된 이미지 데이터이다.

뉴럴네트워크의 하이퍼파라미터는 여러가지가 있으며, 대표적인 하이퍼파라미터는 다음과 같다.

1. 은닉층 개수
2. 은닉노드 개수
3. Activation function
4. Batch size
5. Learning rate
6. Validation score 가 몇 번 동안 개선되지 않았을 때 학습을 종료할 것인지 결정

등이 뉴럴네트워크의 대표적인 하이퍼파라미터이다. 하이퍼파라미터를 여러 번 바꿔보며 비교해보았다.

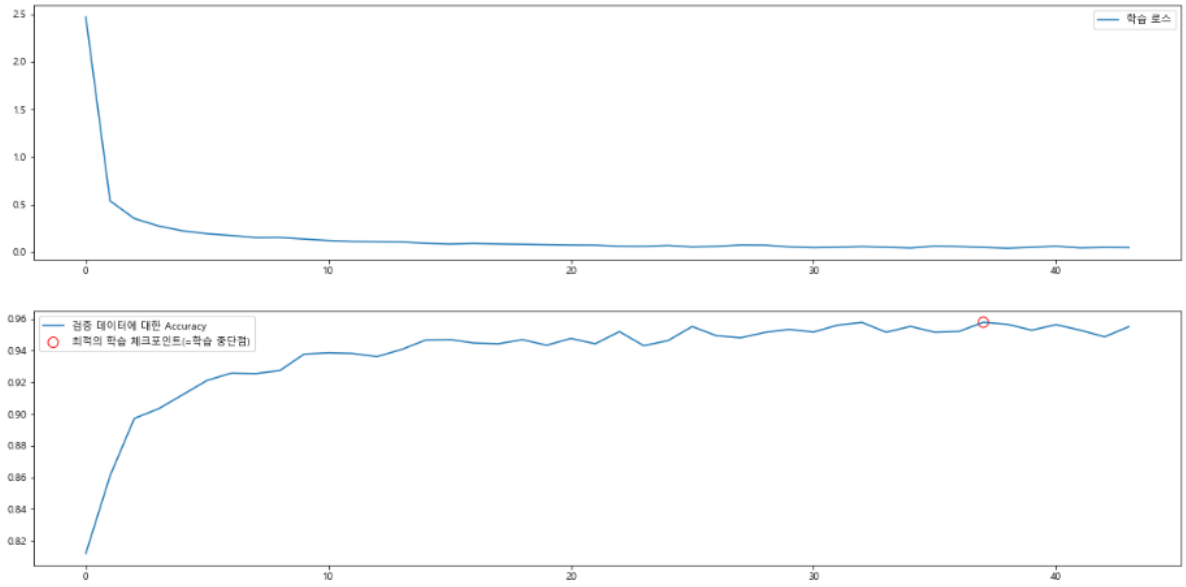
총 데이터 수가 70000 개이기 때문에 10000 개를 testing data 로 분리하고, 60000 개 데이터 중 50000 개를 training data, 10000 개를 validation 데이터로 분리했다. Training data 는 모델은 구축하는 데에만 사용되고, 파라미터 튜닝에 사용되는 데이터는 validation data 이다. 이후 training data 를 통해 위 하이퍼파라미터를 따르고, 입력값은 28X28=784 개의 그레이스케일을 가지고 있기 때문에 입력변수는 총 784 개, 즉 입력 노드는 784 개가 필요하고, 클래스가 10 개이기 때문에 출력 노드는 10 개이다. 이를 통해 뉴럴네트워크 모델을 구축하고, 학습시켜보았다.

1) 은닉층 2 개, 은닉노드 개수(300, 150), activation function=Relu, batch size=100, learning rate=0.001, 5 번의 epoch 이상 validation score 가 개선되지 않을 경우 학습 종료

```
Iteration 1, loss = 2.46856991
Validation score: 0.811900
Iteration 2, loss = 0.53939934
Validation score: 0.861000
Iteration 3, loss = 0.35378653
Validation score: 0.897200
Iteration 4, loss = 0.27548213
Validation score: 0.903300
Iteration 5, loss = 0.22430878
Validation score: 0.912200
Iteration 6, loss = 0.19429326
Validation score: 0.921200
Iteration 7, loss = 0.17596688
Validation score: 0.925800
Iteration 8, loss = 0.15331741
Validation score: 0.925500
Iteration 9, loss = 0.15508077
Validation score: 0.927400
Iteration 10, loss = 0.13827097
Validation score: 0.937800
Iteration 11, loss = 0.12281993
Validation score: 0.938600
Iteration 12, loss = 0.11291188
Validation score: 0.938200
Iteration 13, loss = 0.11113474
Validation score: 0.936200
Iteration 14, loss = 0.10879645
Validation score: 0.940600
Iteration 15, loss = 0.09362884
Validation score: 0.946600
Iteration 16, loss = 0.08649572
Validation score: 0.947000
Iteration 17, loss = 0.09217169
Validation score: 0.944800
Iteration 18, loss = 0.08687312
Validation score: 0.944300
Iteration 19, loss = 0.08337202
Validation score: 0.946900
Iteration 20, loss = 0.07943648
Validation score: 0.943400
Iteration 21, loss = 0.07592185
Validation score: 0.947700
Iteration 22, loss = 0.07421829
Validation score: 0.944300
Iteration 23, loss = 0.06311534
Validation score: 0.952000
Iteration 24, loss = 0.06183623
Validation score: 0.943100
Iteration 25, loss = 0.07107214
Validation score: 0.946300
Iteration 26, loss = 0.05626869
Validation score: 0.955200
Iteration 27, loss = 0.06195116
Validation score: 0.949400
Iteration 28, loss = 0.07609600
Validation score: 0.948100
Iteration 29, loss = 0.07439839
Validation score: 0.951500
Iteration 30, loss = 0.05677506
Validation score: 0.953400
Iteration 31, loss = 0.04866369
Validation score: 0.951700
Iteration 32, loss = 0.05271488
Validation score: 0.956100
Iteration 33, loss = 0.05982919
Validation score: 0.957800
Iteration 34, loss = 0.05296947
Validation score: 0.951600
Iteration 35, loss = 0.04469992
Validation score: 0.955300
Iteration 36, loss = 0.06533005
Validation score: 0.951600
Iteration 37, loss = 0.06015244
Validation score: 0.952100
Iteration 38, loss = 0.05140406
Validation score: 0.957900
Iteration 39, loss = 0.04192649
Validation score: 0.956600
Iteration 40, loss = 0.05270741
Validation score: 0.952900
Iteration 41, loss = 0.06376416
Validation score: 0.956400
Iteration 42, loss = 0.04594184
Validation score: 0.952900
Iteration 43, loss = 0.05129246
Validation score: 0.948700
Iteration 44, loss = 0.04908590
Validation score: 0.955100
```

모델학습시간 : 476 초

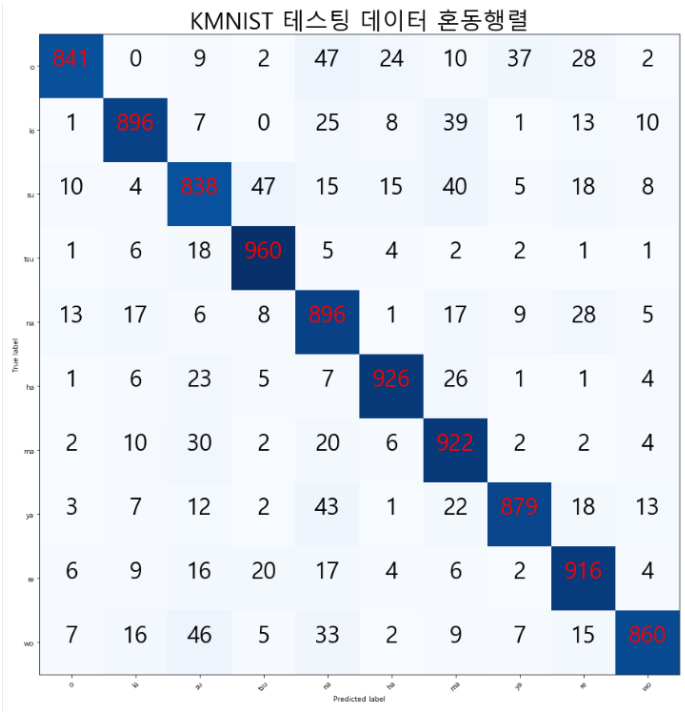
Iteration 을 거듭할수록 train loss 값이 감소했음을 알 수 있고, overfitting 을 방지하기 위해 early stopping 되었다.



(37, 0.9579) 최적의 학습 체크포인트는 37 번째 iteration(epoch)이고 이때 validation accuracy 는 0.9579 이다.

```
훈련 데이터셋 정확도: 0.986 | 테스트용 데이터셋 정확도: 0.893
```

Training accuracy 와 testing accuracy 도 계산해보았다. Training accuracy 는 0.986 으로 매우 높지만, testing accuracy 는 그보다 0.1 정도 낮은 0.893 을 기록했다.



실제 y 값과 예측된 y 값이 일치했는지 나타내는 혼동행렬을 구해보았다. 각 클래스에 해당하는 손글씨들이 꽤 높은 정확도로 분류되었음을 알 수 있다.

Testing data 중 일부 100 개의 데이터를 뽑아 분류 패턴을 확인해보았다. Probability 가 0.13 으로 나와 잘 못 분류된 것도 있지만, 대부분 정확히 분류되었음을 알 수 있다.

2) 은닉층 2 개, 은닉노드 개수(300, 150), activation function=Relu, batch size=500, learning rate=0.001, 5 번의 epoch 이상 validation score 가 개선되지 않을 경우 학습 종료

Batch size 가 커졌으므로 학습 속도는 증가하고, 성능은 감소될 것으로 예상했다. Batch size 가 500 으로 증가했을 때 뉴럴네트워크 모델을 구축하고 정확도를 비교해보았다.

```
Iteration 1, loss = 6.63994304
Validation score: 0.812100
Iteration 2, loss = 1.22713269
Validation score: 0.807500
Iteration 3, loss = 0.63909989
Validation score: 0.850500
Iteration 4, loss = 0.42357188
Validation score: 0.862100
Iteration 5, loss = 0.29796205
Validation score: 0.870800
Iteration 6, loss = 0.22066486
Validation score: 0.883400
Iteration 7, loss = 0.16892839
Validation score: 0.887000
Iteration 8, loss = 0.12890018
Validation score: 0.896700
Iteration 9, loss = 0.10291574
Validation score: 0.896600
Iteration 10, loss = 0.07885155
Validation score: 0.900800
Iteration 11, loss = 0.06124235
Validation score: 0.903000
Iteration 12, loss = 0.05145409
Validation score: 0.905700
Iteration 13, loss = 0.04028497
Validation score: 0.906400
Iteration 14, loss = 0.03204789
Validation score: 0.907400
Iteration 15, loss = 0.02490620
Validation score: 0.906800
Iteration 16, loss = 0.01950687
Validation score: 0.907600
Iteration 17, loss = 0.01750622
Validation score: 0.912300
Iteration 18, loss = 0.01452100
Validation score: 0.913900
Iteration 19, loss = 0.01334254
Validation score: 0.910200
Iteration 20, loss = 0.02118142
Validation score: 0.910300
Iteration 21, loss = 0.04892450
Validation score: 0.908100
Iteration 22, loss = 0.11510945
Validation score: 0.906300
Iteration 23, loss = 0.13750708
Validation score: 0.910900
Iteration 24, loss = 0.09916096
Validation score: 0.921500
Iteration 25, loss = 0.07487706
Validation score: 0.922800
Iteration 26, loss = 0.06811022
Validation score: 0.927600
Iteration 27, loss = 0.04086021
Validation score: 0.928300
Iteration 28, loss = 0.03983214
Validation score: 0.932200
Iteration 29, loss = 0.03625093
Validation score: 0.929100
Iteration 30, loss = 0.02270600
Validation score: 0.931500
Iteration 31, loss = 0.01987070
Validation score: 0.932200
Iteration 32, loss = 0.02606544
Validation score: 0.932800
Iteration 33, loss = 0.03016023
Validation score: 0.930700
Iteration 34, loss = 0.04983517
Validation score: 0.933000
Iteration 35, loss = 0.05881186
Validation score: 0.929800
Iteration 36, loss = 0.06666202
Validation score: 0.932200
Iteration 37, loss = 0.06627845
Validation score: 0.927400
Iteration 38, loss = 0.05798178
Validation score: 0.932800
Iteration 39, loss = 0.05933932
Validation score: 0.936600
Iteration 40, loss = 0.04575202
Validation score: 0.933900
Iteration 41, loss = 0.04560978
Validation score: 0.935900
Iteration 42, loss = 0.03874145
Validation score: 0.940100
Iteration 43, loss = 0.03943415
Validation score: 0.937700
Iteration 44, loss = 0.03811870
Validation score: 0.941000
Iteration 45, loss = 0.03856883
Validation score: 0.938800
Iteration 46, loss = 0.04720778
Validation score: 0.938100
Iteration 47, loss = 0.04431804
Validation score: 0.937900
Iteration 48, loss = 0.03899135
Validation score: 0.943100
Iteration 49, loss = 0.04086932
Validation score: 0.938200
Iteration 50, loss = 0.04921242
Validation score: 0.940700
Iteration 51, loss = 0.04422744
Validation score: 0.940100
Iteration 52, loss = 0.05618630
Validation score: 0.944600
Iteration 53, loss = 0.04596581
Validation score: 0.945200
Iteration 54, loss = 0.02548197
Validation score: 0.944700
Iteration 55, loss = 0.02417724
Validation score: 0.943700
Iteration 56, loss = 0.03728518
Validation score: 0.946100
Iteration 57, loss = 0.03108779
Validation score: 0.942200
Iteration 58, loss = 0.04844759
Validation score: 0.944200
Iteration 59, loss = 0.03853075
Validation score: 0.946300
Iteration 60, loss = 0.03630699
Validation score: 0.946100
Iteration 61, loss = 0.03535845
Validation score: 0.947400
Iteration 62, loss = 0.02510605
Validation score: 0.947700
Iteration 63, loss = 0.02659317
Validation score: 0.945600
Iteration 64, loss = 0.03831787
Validation score: 0.946000
Iteration 65, loss = 0.03129828
Validation score: 0.947800
Iteration 66, loss = 0.05166542
Validation score: 0.942200
```

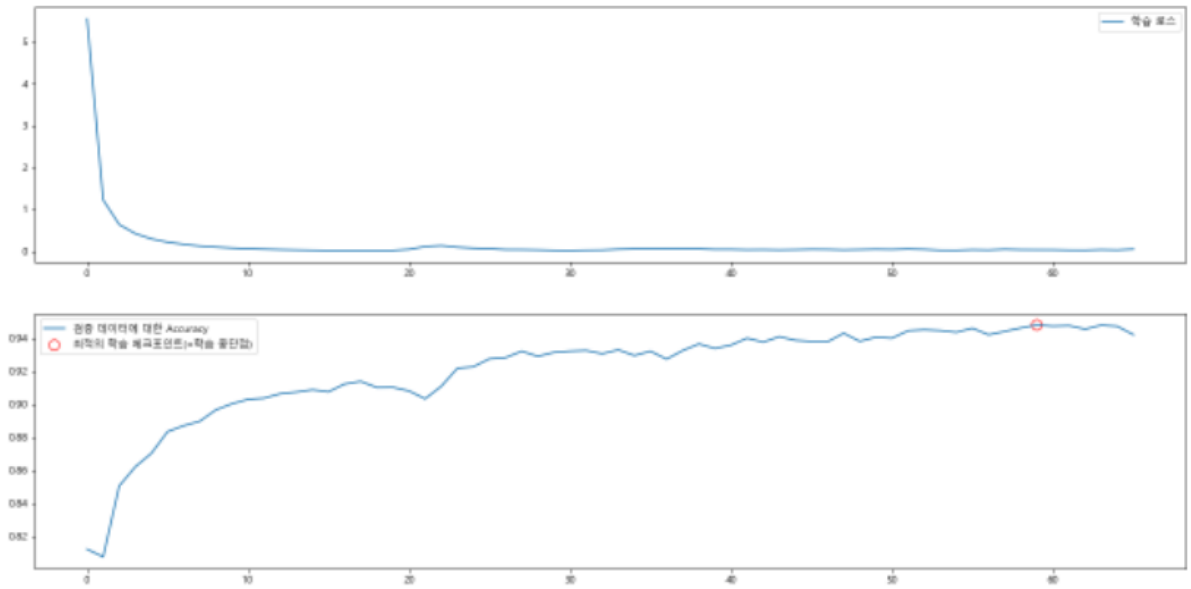
Iteration 을 거듭할수록 train loss 값이 감소했음을 알 수 있고, overfitting 을 방지하기 위해 early stopping 되었다.

모델학습시간 : 268 초

학습속도는 case 1 에 비해 상당히 많이 증가했다.

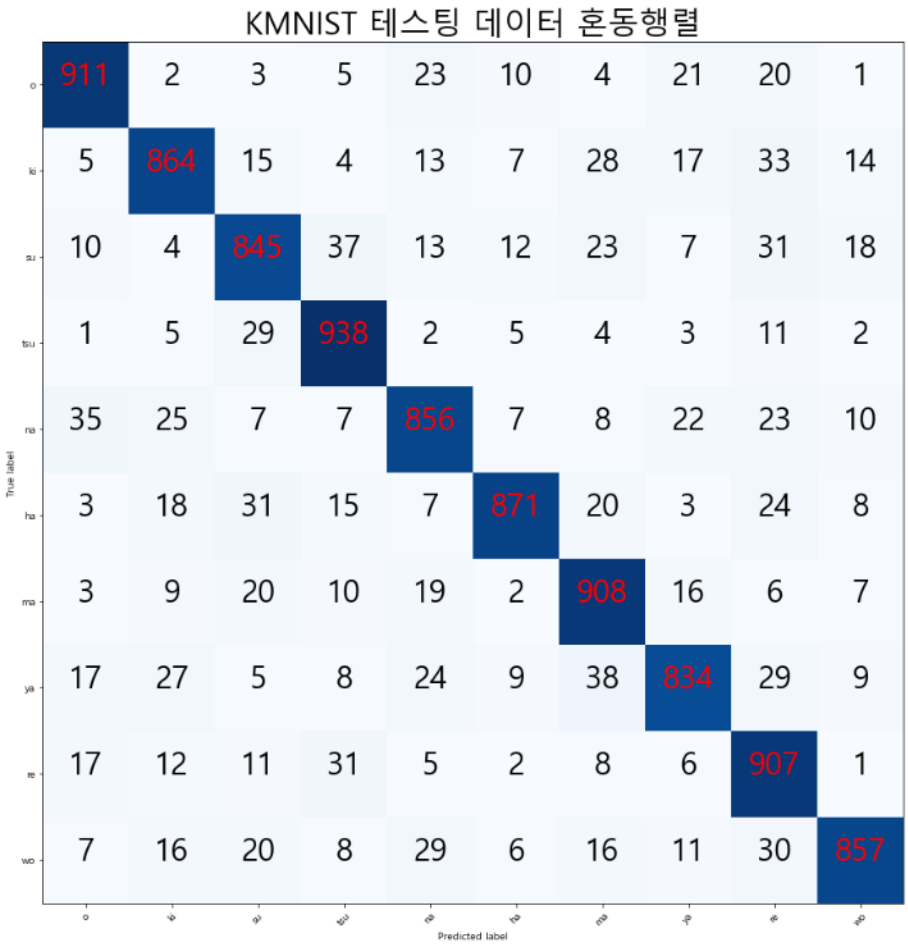
(59, 0.9481)

최적의 학습 체크포인트는 59 번째 iteration(epoch)이고 이때 validation accuracy 는 0.9481 이다.



Training accuracy 와 testing accuracy 도 계산해보았다. Training accuracy 는 0.986 으로 높은 편이지만, testing accuracy 는 그보다 0.1 정도 낮은 0.879 을 기록했다.

```
훈련 데이터셋 정확도: 0.986 | 테스트용 데이터셋 정확도: 0.879
```



실제 y 값과 예측된 y 값이 일치했는지 나타내는 혼동행렬을 구해보았다. 각 클래스에 해당하는 손글씨들이 꽤 높은 정확도로 분류되었음을 알 수 있다.

Testing data 중 일부 100 개의 데이터를 뽑아 분류 패턴을 확인해보았다. Probability 가 0.13 으로 나와 잘 못 분류된 것도 있지만, 대부분 정확히 분류되었음을 알 수 있다.

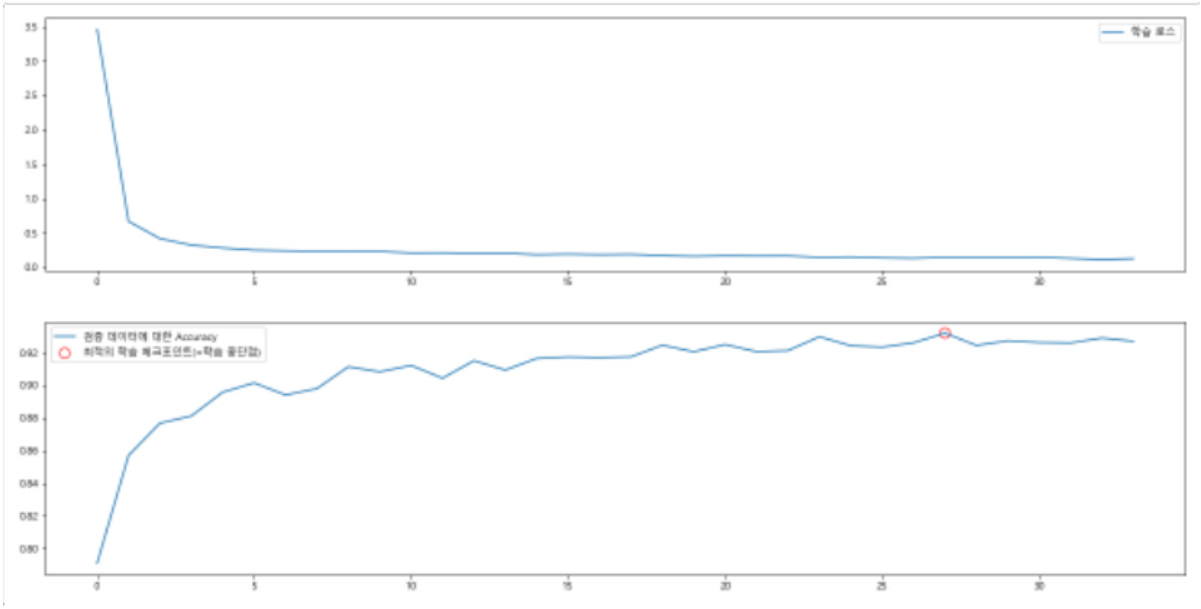
3) 은닉층 1 개, 은닉노드 개수(300), activation function=Relu, batch size=100, learning rate=0.001, 5 번의 epoch 이상 validation score 가 개선되지 않을 경우 학습 종료

은닉층이 줄었으므로 모델이 단순해졌기 때문에 학습 속도가 증가할 것이고, training accuracy 가 감소할 것으로 예측했다. Case 1 에 비해 은닉층이 감소한 것이기 때문에 만약 case 1 이 overfitting 에 가까운 상태였다면 testing accuracy 가 증가할 것이고, case1 이 underfitting 에 가까운 상태였다면 testing accuracy 가 감소할 것으로 예측했다.

```
Iteration 1, loss = 3.46052782
Validation score: 0.791000
Iteration 2, loss = 0.67279744
Validation score: 0.857000
Iteration 3, loss = 0.42216774
Validation score: 0.876800
Iteration 4, loss = 0.32808932
Validation score: 0.881200
Iteration 5, loss = 0.28595670
Validation score: 0.895800
Iteration 6, loss = 0.25513025
Validation score: 0.901500
Iteration 7, loss = 0.24487994
Validation score: 0.894100
Iteration 8, loss = 0.23747996
Validation score: 0.898000
Iteration 9, loss = 0.24167938
Validation score: 0.911400
Iteration 10, loss = 0.23639512
Validation score: 0.908300
Iteration 11, loss = 0.21338795
Validation score: 0.912200
Iteration 12, loss = 0.21498307
Validation score: 0.904400
Iteration 13, loss = 0.20887699
Validation score: 0.915100
Iteration 14, loss = 0.21184936
Validation score: 0.909400
Iteration 15, loss = 0.18822566
Validation score: 0.916500
Iteration 16, loss = 0.19617062
Validation score: 0.917400
Iteration 17, loss = 0.18813973
Validation score: 0.917000
Iteration 18, loss = 0.19303051
Validation score: 0.917500
Iteration 19, loss = 0.17498478
Validation score: 0.924600
Iteration 20, loss = 0.16492410
Validation score: 0.920600
Iteration 21, loss = 0.17286743
Validation score: 0.925000
Iteration 22, loss = 0.17030223
Validation score: 0.920700
Iteration 23, loss = 0.17161138
Validation score: 0.921300
Iteration 24, loss = 0.14930860
Validation score: 0.929800
Iteration 25, loss = 0.15459457
Validation score: 0.924300
Iteration 26, loss = 0.14227715
Validation score: 0.923400
Iteration 27, loss = 0.13607761
Validation score: 0.926100
Iteration 28, loss = 0.15059180
Validation score: 0.932000
Iteration 29, loss = 0.14535836
Validation score: 0.924500
Iteration 30, loss = 0.14585027
Validation score: 0.927300
Iteration 31, loss = 0.15238217
Validation score: 0.926300
Iteration 32, loss = 0.13541106
Validation score: 0.926000
Iteration 33, loss = 0.11781392
Validation score: 0.929000
Iteration 34, loss = 0.13093863
Validation score: 0.927000
(27, 0.932)
```

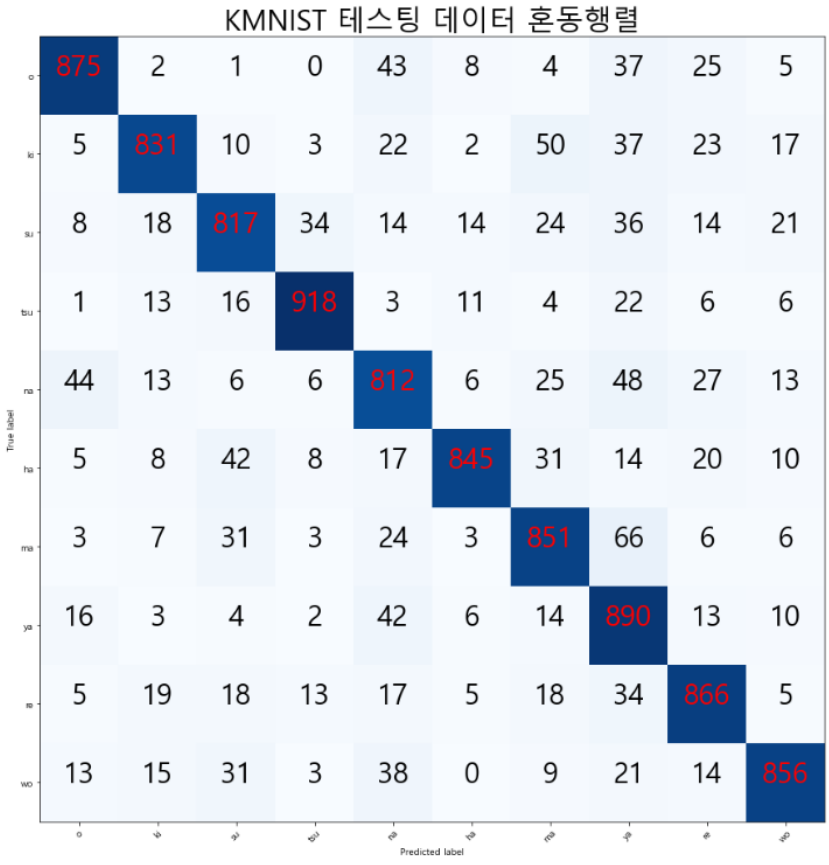
Iteration 을 거듭할수록 train loss 값이 감소했음을 알 수 있고, overfitting 을 방지하기 위해 early stopping 되었다.

최적의 학습 체크포인트는 27 번째 iteration(epoch)이고 이때 validation accuracy 는 0.932 이다.



훈련 데이터셋 정확도: 0.966 | 테스트용 데이터셋 정확도: 0.856

Training accuracy 와 testing accuracy 도 계산해보았다. Training accuracy 는 0.966 으로 높은 편이지만, testing accuracy 는 그보다 0.1 정도 낮은 0.856 을 기록했다. Case1 과 비교했을 때 training accuracy 와 testing accuracy 가 떨어졌는데, 은닉층의 감소로 인해 모델이 underfitting 되었다고 해석했다.



875	2	1	0	43	8	4	37	25	5
5	831	10	3	22	2	50	37	23	17
8	18	817	34	14	14	24	36	14	21
1	13	16	918	3	11	4	22	6	6
44	13	6	6	812	6	25	48	27	13
5	8	42	8	17	845	31	14	20	10
3	7	31	3	24	3	851	66	6	6
16	3	4	2	42	6	14	890	13	10
5	19	18	13	17	5	18	34	866	5
13	15	31	3	38	0	9	21	14	856

실제 y 값과 예측된 y 값이 일치했는지 나타내는 혼동행렬을 구해보았다. 각 클래스에 해당하는 손글씨들이 꽤 높은 정확도로 분류되었음을 알 수 있다.

Testing data 중 일부 100 개의 데이터를 뽑아 분류 패턴을 확인해보았다. Probability 가 0.37 로 나와 잘 못 분류된 것도 있지만, 대부분 정확히 분류되었음을 알 수 있다.

4) 은닉층 2 개, 은닉노드 개수(300, 150), activation function=tanh, batch size=100, learning rate=0.001, 5 번의 epoch 이상 validation score 가 개선되지 않을 경우 학습 종료

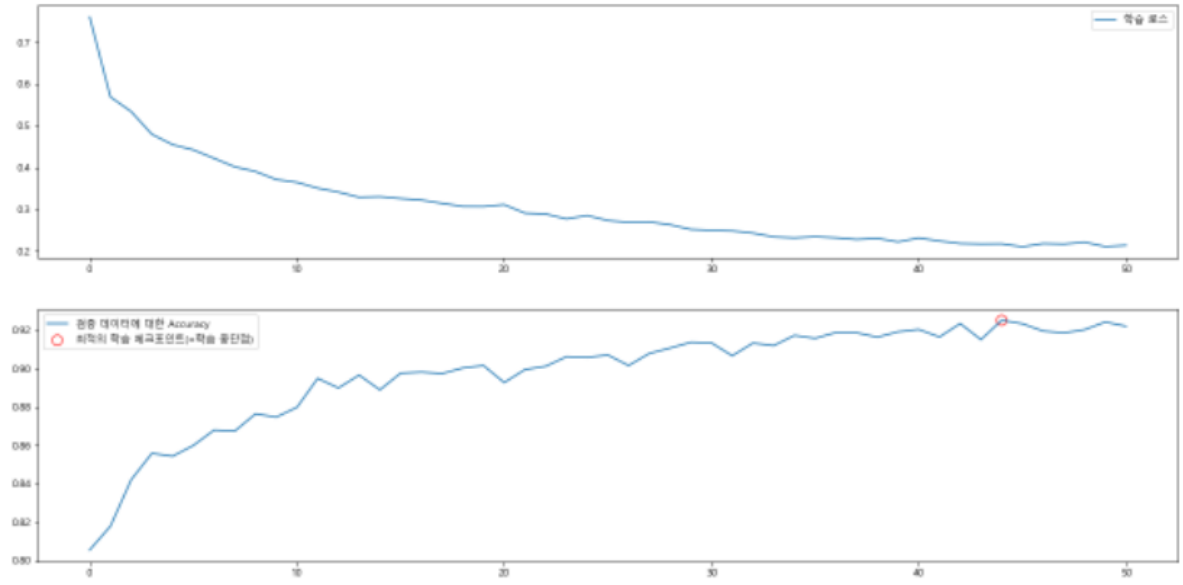
Tanh 함수는 모델이 어느 정도까지만 진화하고, 이후에는 기울기가 거의 0 에 수렴하기 때문에 모델이 더 진화하지 못하는 단점이 있다. 따라서 Relu 함수를 활성화 함수로 사용하는 것에 비해 모델 성능을 떨어질 것으로 예측했다.

```
Iteration 1, loss = 0.76023590
Validation score: 0.805500
Iteration 2, loss = 0.56896451
Validation score: 0.817900
Iteration 3, loss = 0.53396261
Validation score: 0.842100
Iteration 4, loss = 0.47956098
Validation score: 0.855900
Iteration 5, loss = 0.45464238
Validation score: 0.854400
Iteration 6, loss = 0.44220911
Validation score: 0.860000
Iteration 7, loss = 0.42196253
Validation score: 0.868000
Iteration 8, loss = 0.40152788
Validation score: 0.867500
Iteration 9, loss = 0.39035691
Validation score: 0.876500
Iteration 10, loss = 0.37114136
Validation score: 0.874800
Iteration 11, loss = 0.36438739
Validation score: 0.879900
Iteration 12, loss = 0.35041355
Validation score: 0.895000
Iteration 13, loss = 0.34104740
Validation score: 0.890000
Iteration 14, loss = 0.32863082
Validation score: 0.896700
Iteration 15, loss = 0.33001748
Validation score: 0.889000
Iteration 16, loss = 0.32560165
Validation score: 0.897700
Iteration 17, loss = 0.32188281
Validation score: 0.898300
Iteration 18, loss = 0.31412233
Validation score: 0.897500
Iteration 19, loss = 0.30677454
Validation score: 0.900400
Iteration 20, loss = 0.30655455
Validation score: 0.901700
Iteration 21, loss = 0.31085383
Validation score: 0.892800
Iteration 22, loss = 0.29062919
Validation score: 0.899600
Iteration 23, loss = 0.28848086
Validation score: 0.901300
Iteration 24, loss = 0.27690564
Validation score: 0.906300
Iteration 25, loss = 0.28494410
Validation score: 0.905900
Iteration 26, loss = 0.27295353
Validation score: 0.907200
Iteration 27, loss = 0.26844408
Validation score: 0.901700
Iteration 28, loss = 0.26962004
Validation score: 0.908000
Iteration 29, loss = 0.26286333
Validation score: 0.910800
Iteration 30, loss = 0.25167255
Validation score: 0.913800
Iteration 31, loss = 0.24938969
Validation score: 0.913500
Iteration 32, loss = 0.24812319
Validation score: 0.906800
Iteration 33, loss = 0.24277078
Validation score: 0.913400
Iteration 34, loss = 0.23375864
Validation score: 0.912100
Iteration 35, loss = 0.23136200
Validation score: 0.917400
Iteration 36, loss = 0.23440021
Validation score: 0.915900
Iteration 37, loss = 0.23159249
Validation score: 0.918900
Iteration 38, loss = 0.22762458
Validation score: 0.918800
Iteration 39, loss = 0.23025909
Validation score: 0.916500
Iteration 40, loss = 0.22179010
Validation score: 0.919300
Iteration 41, loss = 0.23107016
Validation score: 0.920500
Iteration 42, loss = 0.22387873
Validation score: 0.916600
Iteration 43, loss = 0.21785365
Validation score: 0.923600
Iteration 44, loss = 0.21671425
Validation score: 0.915200
Iteration 45, loss = 0.21706856
Validation score: 0.925300
Iteration 46, loss = 0.21002987
Validation score: 0.923500
Iteration 47, loss = 0.21769544
Validation score: 0.919700
Iteration 48, loss = 0.21647304
Validation score: 0.918700
Iteration 49, loss = 0.22101335
Validation score: 0.920400
Iteration 50, loss = 0.21044088
Validation score: 0.924400
Iteration 51, loss = 0.21347324
Validation score: 0.922200
```

Iteration 을 거듭할수록 train loss 값이 감소했음을 알 수 있고, overfitting 을 방지하기 위해 early stopping 되었다.

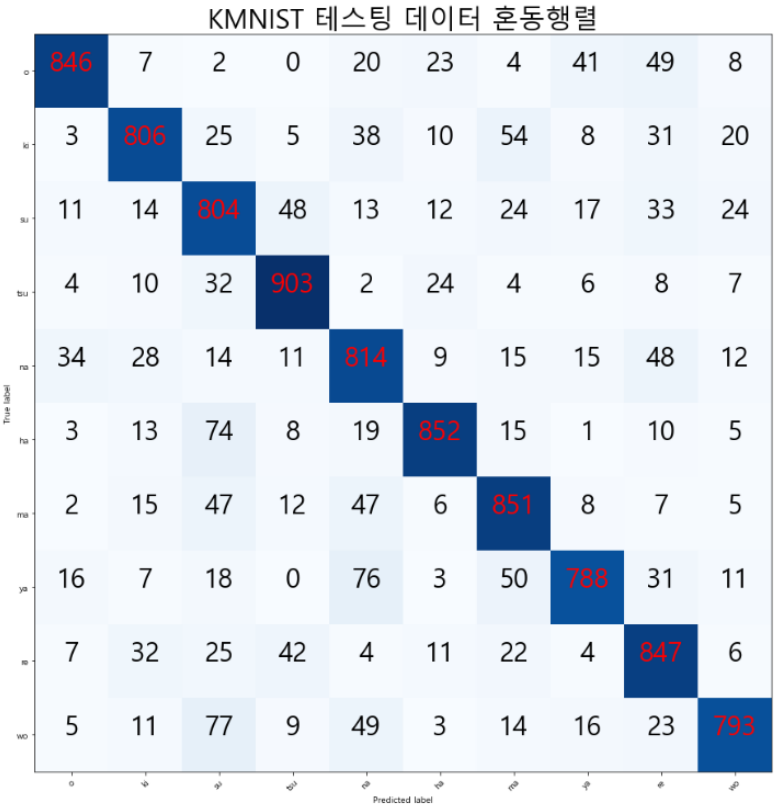
```
(44, 0.9253)
```

최적의 학습 체크포인트는 44 번째 iteration(epoch)이고 이때 validation accuracy 는 0.9253 이다.



훈련 데이터셋 정확도: 0.933 | 테스트용 데이터셋 정확도: 0.830

Training accuracy 와 testing accuracy 도 계산해보았다. Training accuracy 는 0.933 으로 괜찮은 편이지만, testing accuracy 는 그보다 0.1 정도 낮은 0.830 을 기록했다. 나쁘지 않은 수치라고 볼 수 있지만, Relu 활성화함수를 사용했을 때 대비 0.05 정도 정확도가 떨어졌기 때문에 상당히 많이 정확도가 감소했다고 할 수 있다. 따라서 tanh 함수는 Relu 함수에 비해 성능이 많이 떨어진다고 해석했다.



실제 y 값과 예측된 y 값이 일치했는지 나타내는 혼동행렬을 구해보았다. 대각선에 있는 요소 이외에 잘못 분류된 값들을 이전 case 들과 비교해봤을 때, 짙은 파랑색을 나타내는 square 들이 늘어났음을 알 수 있다.

Testing data 중 일부 100 개의 데이터를 뽑아 분류 패턴을 확인해보았다. Probability 가 0.39, 0.4 등으로 나와 잘 못 분류된 것들이 일부 보였다.

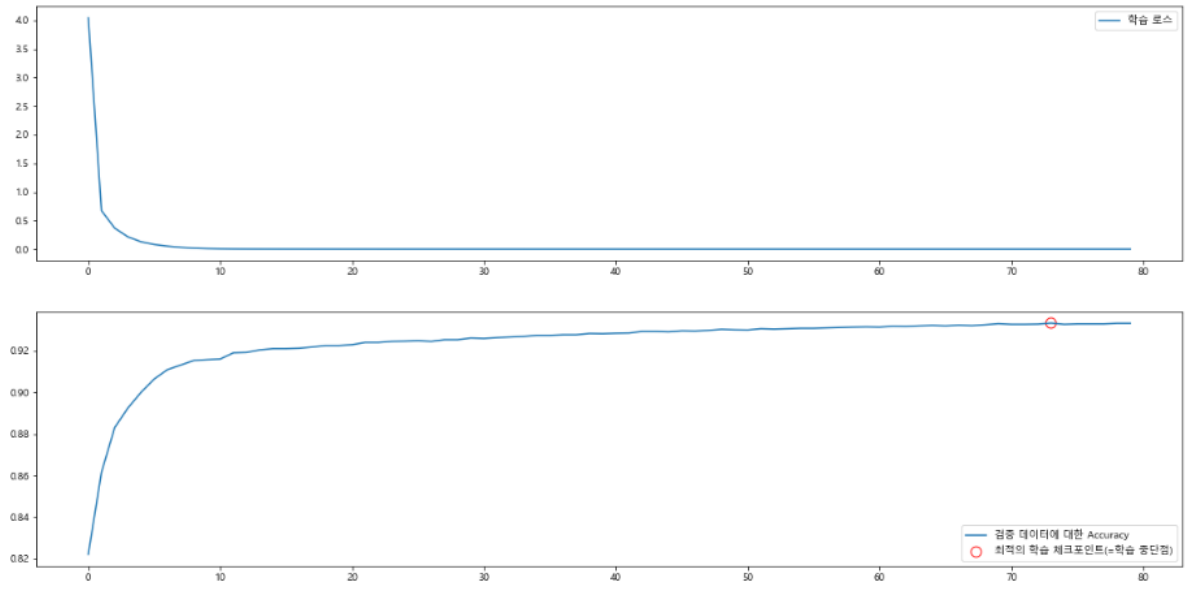
5) 은닉층 3 개, 은닉노드 개수(450, 300, 150), activation function=tanh, batch size=100, learning rate=0.001, 5 번의 epoch 이상 validation score 가 개선되지 않을 경우 학습 종료

은닉층이 증가했고 은닉노드 수도 증가했기 때문에 모델이 복잡해졌다. 따라서 학습 속도가 감소할 것이고, training accuracy 가 증가할 것으로 예측했다. Case 1 에 비해 모델이 복잡해졌기 때문에 만약 case 1 이 overfitting 에 가까운 상태였다면 testing accuracy 가 감소할 것이고, case1 이 underfitting 에 가까운 상태였다면 testing accuracy 가 증가할 것으로 예측했다.

Iteration 을 거듭할수록 train loss 값이 감소했음을 알 수 있고, overfitting 을 방지하기 위해 early stopping 되었다.

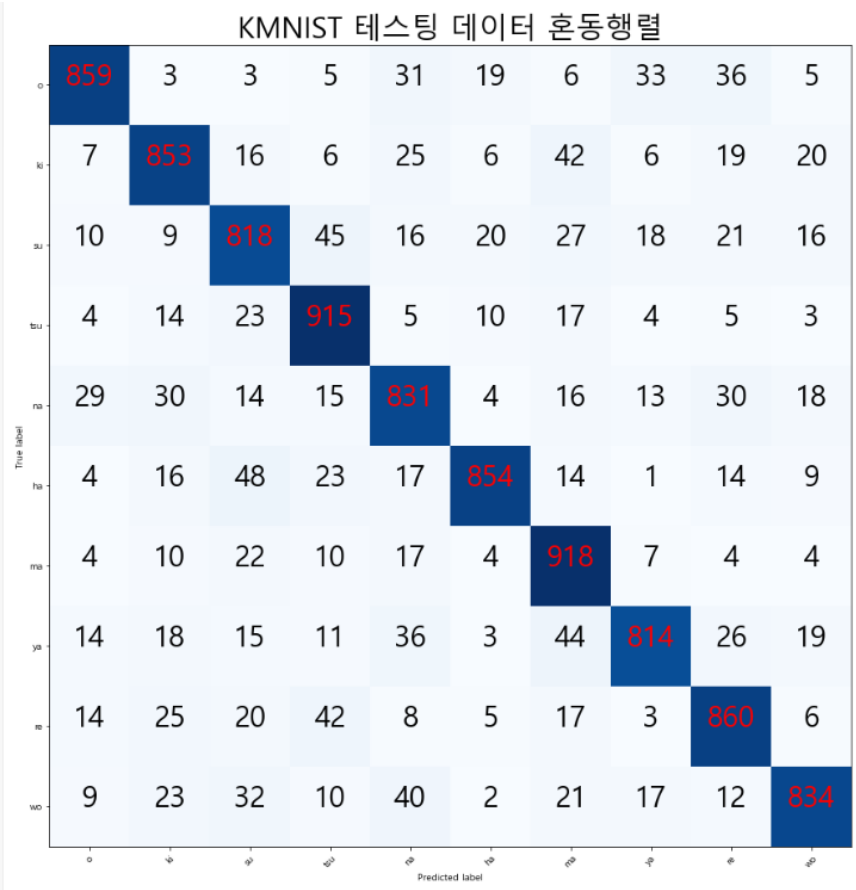
(73, 0.9334) 최적의 학습 체크포인트는 73 번째 iteration(epoch)이고 이때 validation accuracy 는 0.9334 이다.

```
Iteration 1, loss = 4.04499599
Validation score: 0.822100
Iteration 2, loss = 0.67808784
Validation score: 0.881600
Iteration 3, loss = 0.38725482
Validation score: 0.888100
Iteration 4, loss = 0.21492068
Validation score: 0.892500
Iteration 5, loss = 0.12809523
Validation score: 0.900100
Iteration 6, loss = 0.08102793
Validation score: 0.906500
Iteration 7, loss = 0.05180434
Validation score: 0.911000
Iteration 8, loss = 0.03073480
Validation score: 0.913200
Iteration 9, loss = 0.01725480
Validation score: 0.915400
Iteration 10, loss = 0.00952750
Validation score: 0.915800
Iteration 11, loss = 0.00568882
Validation score: 0.916100
Iteration 12, loss = 0.00350039
Validation score: 0.919100
Iteration 13, loss = 0.00238578
Validation score: 0.919800
Iteration 14, loss = 0.00214082
Validation score: 0.920500
Iteration 15, loss = 0.00180868
Validation score: 0.921100
Iteration 16, loss = 0.00156998
Validation score: 0.921100
Iteration 17, loss = 0.00138548
Validation score: 0.921800
Iteration 18, loss = 0.00126711
Validation score: 0.921900
Iteration 19, loss = 0.00111658
Validation score: 0.922800
Iteration 20, loss = 0.00101424
Validation score: 0.922800
Iteration 21, loss = 0.00092488
Validation score: 0.923000
Iteration 22, loss = 0.00084484
Validation score: 0.924100
Iteration 23, loss = 0.00077981
Validation score: 0.924100
Iteration 24, loss = 0.00072188
Validation score: 0.924600
Iteration 25, loss = 0.00067088
Validation score: 0.924700
Iteration 26, loss = 0.00062585
Validation score: 0.924900
Iteration 27, loss = 0.00058444
Validation score: 0.924800
Iteration 28, loss = 0.00054749
Validation score: 0.925400
Iteration 29, loss = 0.00051396
Validation score: 0.925400
Iteration 30, loss = 0.00048606
Validation score: 0.926200
Iteration 31, loss = 0.00045671
Validation score: 0.926000
Iteration 32, loss = 0.00043158
Validation score: 0.926400
Iteration 33, loss = 0.00040981
Validation score: 0.926700
Iteration 34, loss = 0.00038866
Validation score: 0.927000
Iteration 35, loss = 0.00036996
Validation score: 0.927600
Iteration 36, loss = 0.00035240
Validation score: 0.927500
Iteration 37, loss = 0.00033621
Validation score: 0.927800
Iteration 38, loss = 0.00032172
Validation score: 0.927800
Iteration 39, loss = 0.00030855
Validation score: 0.928400
Iteration 40, loss = 0.00029584
Validation score: 0.928300
Iteration 41, loss = 0.00028420
Validation score: 0.928500
Iteration 42, loss = 0.00027357
Validation score: 0.928600
Iteration 43, loss = 0.00026380
Validation score: 0.928400
Iteration 44, loss = 0.00025419
Validation score: 0.928400
Iteration 45, loss = 0.00024580
Validation score: 0.928200
Iteration 46, loss = 0.00023795
Validation score: 0.929700
Iteration 47, loss = 0.00023082
Validation score: 0.929600
Iteration 48, loss = 0.00022375
Validation score: 0.929900
Iteration 49, loss = 0.00021702
Validation score: 0.930400
Iteration 50, loss = 0.00021127
Validation score: 0.930200
Iteration 51, loss = 0.00020585
Validation score: 0.930100
Iteration 52, loss = 0.00020049
Validation score: 0.930700
Iteration 53, loss = 0.00019552
Validation score: 0.930500
Iteration 54, loss = 0.00019085
Validation score: 0.930700
Iteration 55, loss = 0.00018653
Validation score: 0.930900
Iteration 56, loss = 0.00018283
Validation score: 0.930900
Iteration 57, loss = 0.00017877
Validation score: 0.931100
Iteration 58, loss = 0.00017515
Validation score: 0.931300
Iteration 59, loss = 0.00017197
Validation score: 0.931500
Iteration 60, loss = 0.00016885
Validation score: 0.931600
Iteration 61, loss = 0.00016582
Validation score: 0.931500
Iteration 62, loss = 0.00016316
Validation score: 0.931900
Iteration 63, loss = 0.00016050
Validation score: 0.931800
Iteration 64, loss = 0.00015804
Validation score: 0.932100
Iteration 65, loss = 0.00015576
Validation score: 0.932600
Iteration 66, loss = 0.00015370
Validation score: 0.932100
Iteration 67, loss = 0.00015166
Validation score: 0.932400
Iteration 68, loss = 0.00014968
Validation score: 0.932200
Iteration 69, loss = 0.00014792
Validation score: 0.932600
Iteration 70, loss = 0.00014625
Validation score: 0.933100
Iteration 71, loss = 0.00014466
Validation score: 0.932800
Iteration 72, loss = 0.00014320
Validation score: 0.932800
Iteration 73, loss = 0.00014179
Validation score: 0.932900
Iteration 74, loss = 0.00014046
Validation score: 0.933400
Iteration 75, loss = 0.00013917
Validation score: 0.932800
Iteration 76, loss = 0.00013798
Validation score: 0.933000
Iteration 77, loss = 0.00013689
Validation score: 0.933000
Iteration 78, loss = 0.00013578
Validation score: 0.933000
Iteration 79, loss = 0.00013482
Validation score: 0.933300
Iteration 80, loss = 0.00013384
Validation score: 0.933300
```



훈련 데이터셋 정확도: 0.989 | 테스트용 데이터셋 정확도: 0.856

Training accuracy 와 testing accuracy 도 계산해보았다. Training accuracy 는 0.989 로 오히려 case1 보다 컸다. 하지만 testing accuracy 는 case1 보다 낮은 0.856 을 기록했다. Training accuracy 는 증가했고, testing accuracy 는 감소했기 때문에 모델이 overfitting 되었다고 해석했다.



859	3	3	5	31	19	6	33	36	5
7	853	16	6	25	6	42	6	19	20
10	9	818	45	16	20	27	18	21	16
4	14	23	915	5	10	17	4	5	3
29	30	14	15	831	4	16	13	30	18
4	16	48	23	17	854	14	1	14	9
4	10	22	10	17	4	918	7	4	4
14	18	15	11	36	3	44	814	26	19
14	25	20	42	8	5	17	3	860	6
9	23	32	10	40	2	21	17	12	834

실제 y 값과 예측된 y 값이 일치했는지 나타내는 혼동행렬을 구해보았다. 각 클래스에 해당하는 손글씨들이 꽤 높은 정확도로 분류되었음을 알 수 있다.

Testing data 중 일부 100 개의 데이터를 뽑아 분류 패턴을 확인해보았다. Probability 가 0.44 로 나와 잘 못 분류된 것도 있지만, 대부분 정확히 분류되었음을 알 수 있다.

종합적으로 평가해보자면 case 1 의 경우가 testing accuracy 가 가장 높았다. 이는 underfitting, overfitting 없이 적당한 복잡도의 모델로 모델링 되었음을 나타내고, 학습 시간도 476 초로 그리 오래 걸리지 않고 성능 저하도 거의 없기 때문에 batch size 도 적당하다고 해석할 수 있다.

3. 감명 깊게 읽었던 책을 명시하고 그 이유를 간단히 쓰시오 (감명 깊게 읽어 던 책 하나 정도는 언제 누가 물어봐도 대답할 준비가 되어 있어야 합니다)

제가 감명 깊게 읽었던 책은 존 스타인벡의 '진주'라는 소설입니다. 읽게 된 계기는 영어 공부를 위해 원서를 구입해서 읽어봤습니다. 간단한 줄거리는, 주인공 키노는 어느날 우연히 커다란 진주를 얻게 되고, 이를 탐낸 주변사람들에 의해 고통받고 쫓기게 됩니다. 결국 고난 끝에 아들을 잃고, 진주를 버리고 다시 집으로 돌아오게 됩니다. 이 책이 감명 깊었던 이유는, 인간의 탐욕을 적나라하게 드러냈기 때문입니다. 진주를 얻게 된 것이 행복의 시작일 줄 알았지만, 불행의 시작이 되었습니다. 악마의 돌임을 알고 몇번을 버리려고 했지만, 결국 키노 본인도 욕망 때문에 버리지 못했고, 결국 가족을 비극으로 몰아넣게 됩니다. 이 책을 읽으면서 인간의 욕심이 가지고 있는 추악함을 적나라하게 묘사하고, 노력없이 얻은 큰 재물에 대한 비판을 가했다는 점이 인상깊었습니다.