Tic-Tac-Toe

인공지능 개인 프로젝트

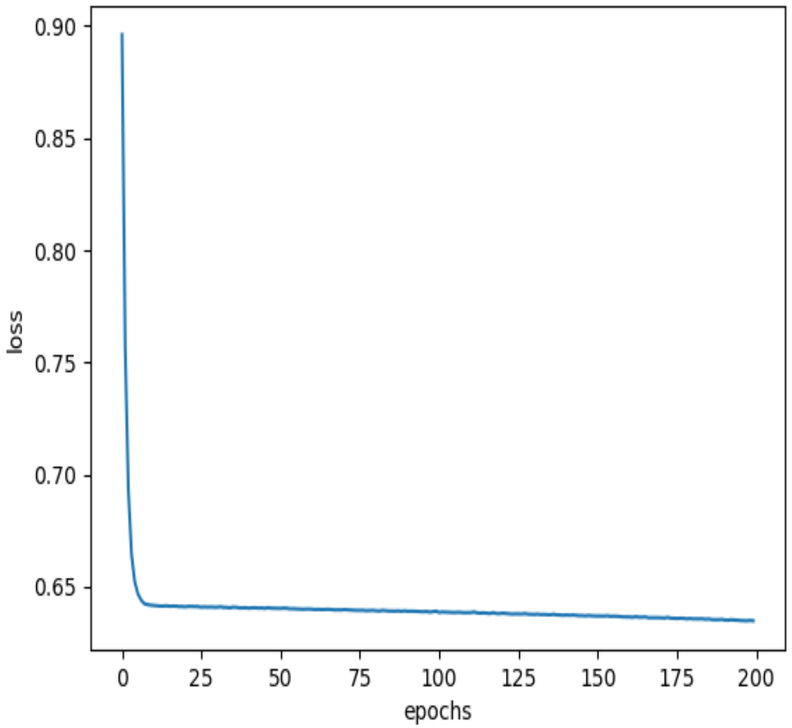
문제 설명

삼목 게임인 Tic Tac Toe를 여러가지 학습방법과
학습률과 epoch를 변동시켜서 딥러닝으로 학습시키고
정확도와 손실률을 측정하고 가장 적절한 방법을
찾아봅니다.

tic-tac-toe.csv의 구성

	A	B	C	D	E	F	G	H	I	J
1	TL	TM	TR	ML	MM	MR	BL	BM	BR	class
2	x	x	x	x	o	o	x	o	o	TRUE
3	x	x	x	x	o	o	o	x	o	TRUE
4	x	x	x	x	o	o	o	o	x	TRUE
5	x	x	x	x	o	o	o	b	b	TRUE
6	x	x	x	x	o	o	b	o	b	TRUE
7	x	x	x	x	o	o	b	b	o	TRUE
8	x	x	x	x	o	b	o	o	b	TRUE
9	x	x	x	x	o	b	o	b	o	TRUE
10	x	x	x	x	o	b	b	o	o	TRUE
11	x	x	x	x	b	o	o	o	b	TRUE
12	x	x	x	x	b	o	o	b	o	TRUE
13	x	x	x	x	b	o	b	o	o	TRUE
14	x	x	x	o	x	o	x	o	o	TRUE
15	x	x	x	o	x	o	o	x	o	TRUE
16	x	x	x	o	x	o	o	o	x	TRUE
17	x	x	x	o	x	o	o	b	b	TRUE
18	x	x	x	o	x	o	b	o	b	TRUE
19	x	x	x	o	x	o	b	b	o	TRUE
20	x	x	x	o	x	b	o	o	b	TRUE
21	x	x	x	o	x	b	o	b	o	TRUE
22	x	x	x	o	x	b	b	o	o	TRUE

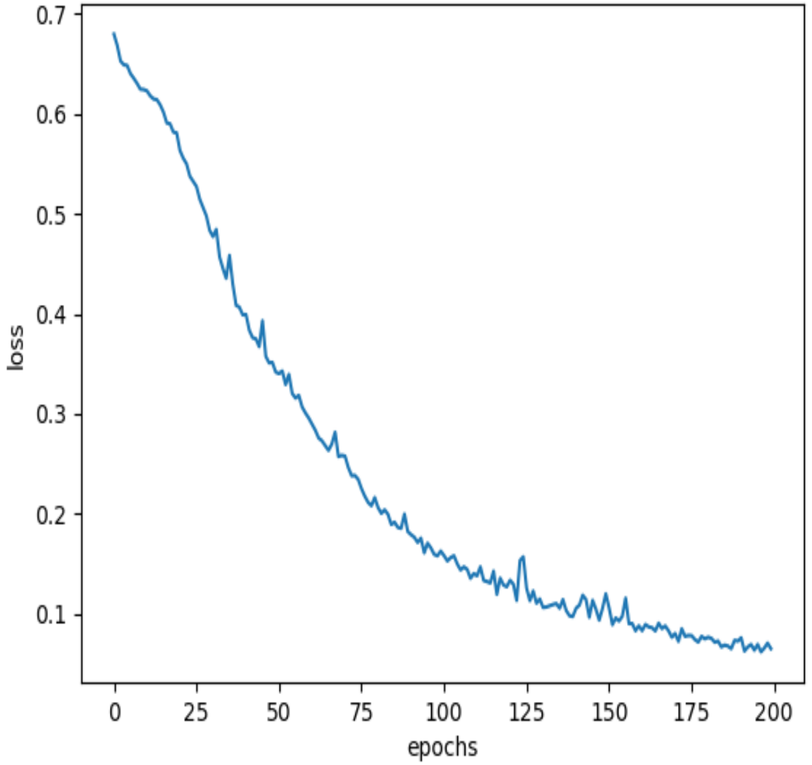
1. SGD




```
len(model.layers): 3
18/18 - 0s - loss: 0.6346 - accuracy: 0.6568
12/12 - 0s - loss: 0.6405 - accuracy: 0.6484
confusion_matrix(C): tf.Tensor(
[[377   0]
 [197   0]], shape=(2, 2), dtype=int32)
```

SGD로 돌렸을 때는 모든 결과를 한쪽으로 분류하는 문제가 발생했습니다.
그래서 정확도는 66%로 덜어지게 되었습니다.

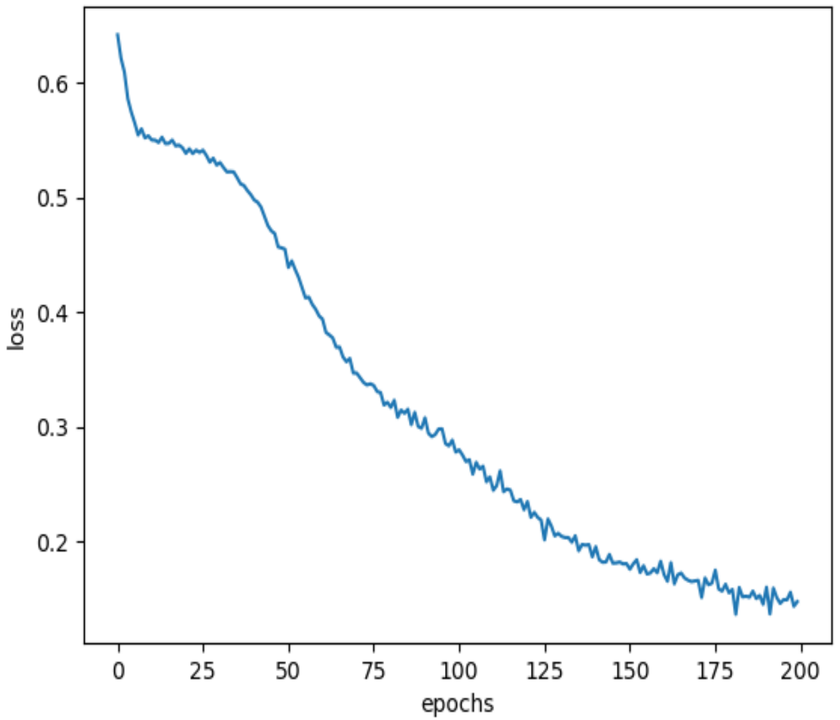
2. ADAM




```
len(model.layers): 3
18/18 - 0s - loss: 0.0898 - accuracy: 0.9704
12/12 - 0s - loss: 0.5783 - accuracy: 0.8594
confusion_matrix(C): tf.Tensor(
[[185  16]
 [  1 372]], shape=(2, 2), dtype=int32)
```

Adam은 약 97%의 정확도를 보였습니다.

3. RMSprop

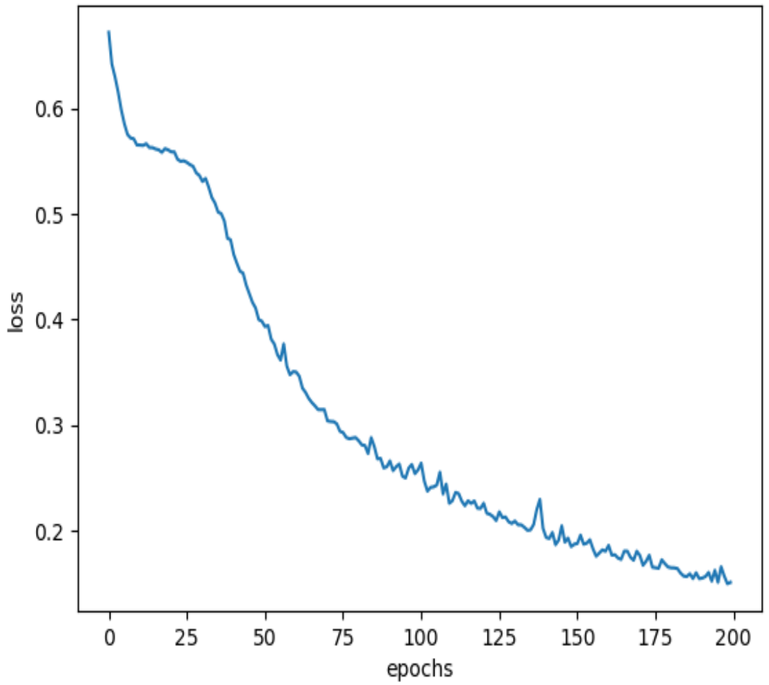



```
len(model.layers): 3
18/18 - 0s - loss: 0.1281 - accuracy: 0.9512
12/12 - 0s - loss: 0.4935 - accuracy: 0.8073
confusion_matrix(C): tf.Tensor(
[[365  15]
 [ 13 181]], shape=(2, 2), dtype=int32)
```

Adam은 약 95%의 정확도를 보였습니다.

ADAM 채택 후
학습률과 epoch에 따른 차이

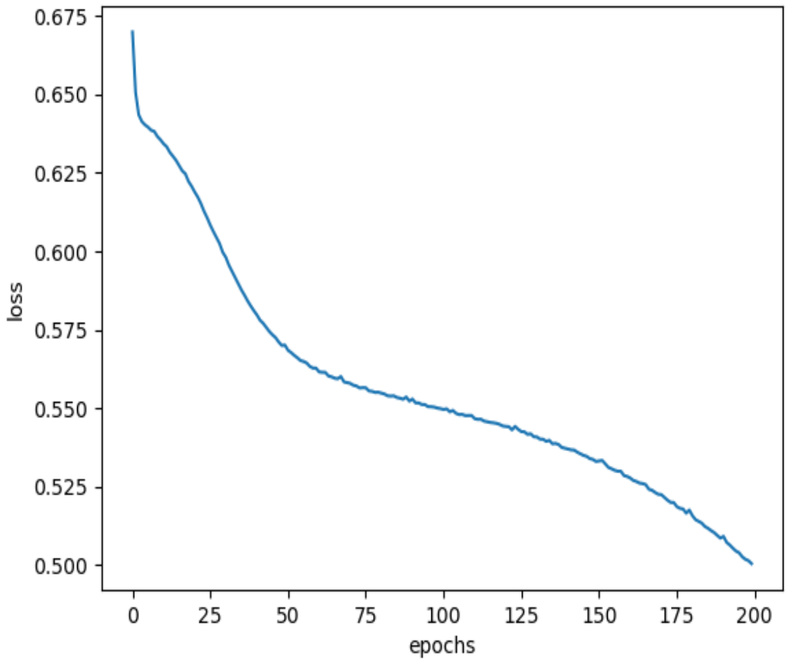
1. 학습률 0.01 / epoch 200번




```
len(model.layers): 3
18/18 - 0s - loss: 0.1394 - accuracy: 0.9582
12/12 - 0s - loss: 0.4783 - accuracy: 0.7995
confusion_matrix(C): tf.Tensor(
[[361  15]
 [  9 189]], shape=(2, 2), dtype=int32)
```

학습률 0.01에 epoches를 200으로 돌렸을 때 약 96%의 정확도를 보였습니다.

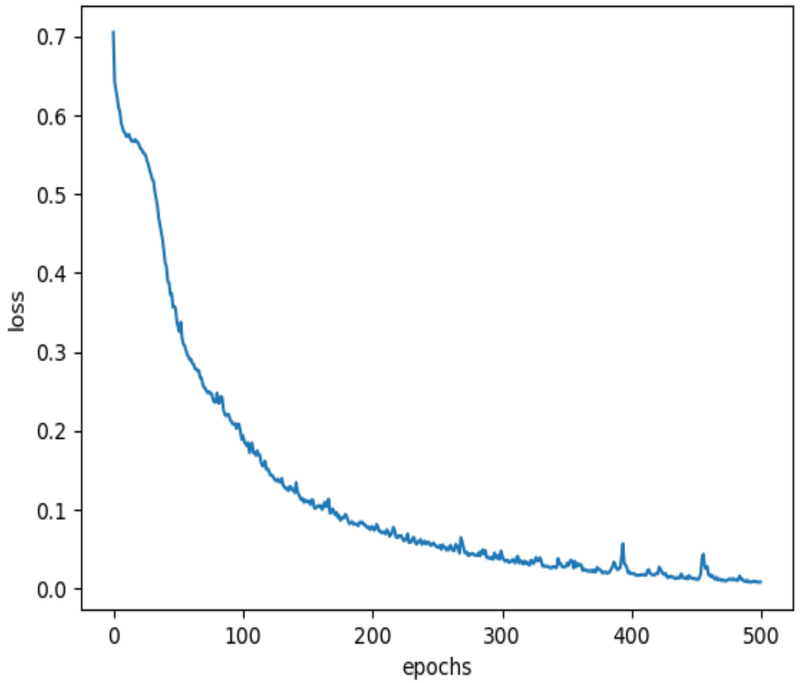
2. 학습률 0.001 / epoch 200번




```
len(model.layers): 3
18/18 - 0s - loss: 0.4994 - accuracy: 0.7544
12/12 - 0s - loss: 0.5638 - accuracy: 0.7031
confusion_matrix(C): tf.Tensor(
[[338  39]
 [102  95]], shape=(2, 2), dtype=int32)
```

학습률 0.001에 epoches를 200으로 돌렸을 때 약 75%의 정확도를 보였습니다.

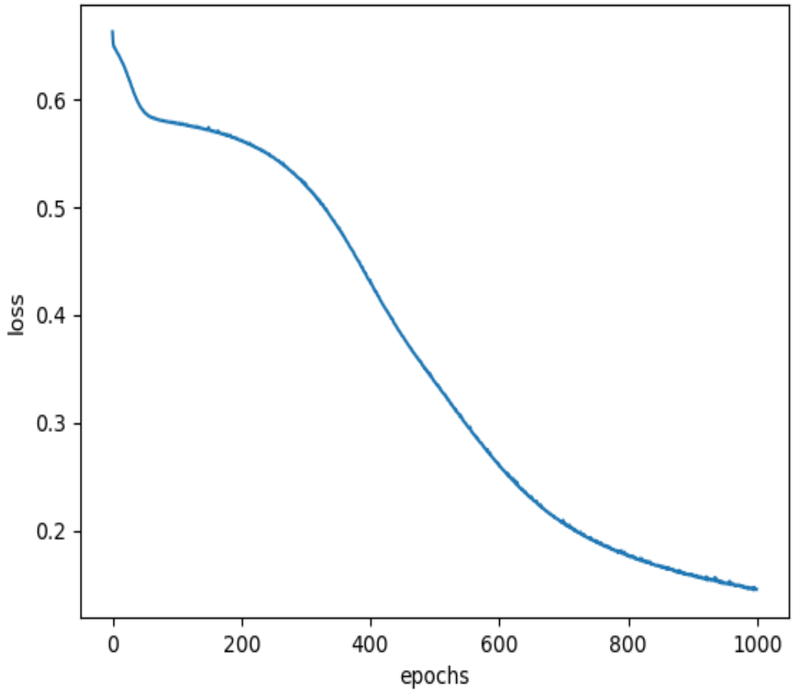
3. 학습률 0.01 / epoch 500번




```
len(model.layers): 3
18/18 - 0s - loss: 0.0069 - accuracy: 1.0000
12/12 - 0s - loss: 2.3830 - accuracy: 0.7812
confusion_matrix(C): tf.Tensor(
[[378   0]
 [  0 196]], shape=(2, 2), dtype=int32)
```

학습률 0.01에 epoches를 500으로 돌렸을 때 약 100%의 정확도를 보였습니다.

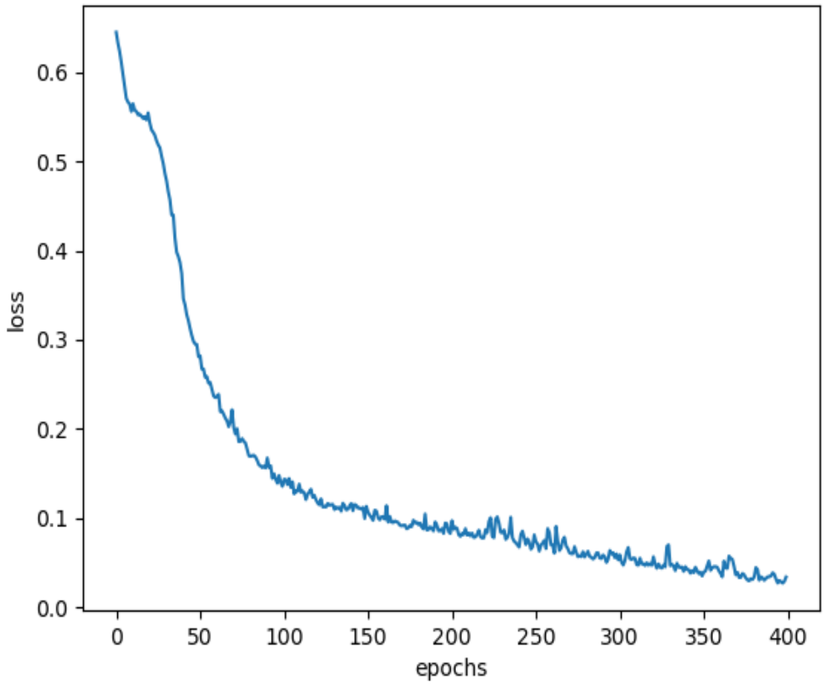
4. 학습률 0.001 / epoch 1000번




```
len(model.layers): 3
18/18 - 0s - loss: 0.1437 - accuracy: 0.9460
12/12 - 0s - loss: 0.3735 - accuracy: 0.8698
confusion_matrix(C): tf.Tensor(
[[354  16]
 [ 15 189]], shape=(2, 2), dtype=int32)
```

학습률 0.001에 epoches를 1000으로 돌렸을 때 약 95%의 정확도를 보였습니다.

5. 학습률 0.01 / epoch 400번

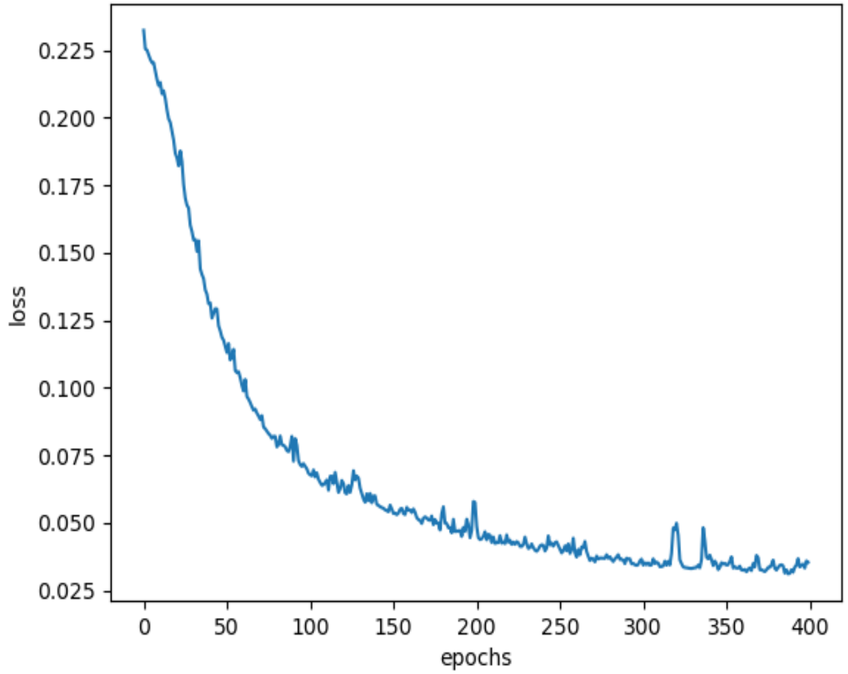



```
len(model.layers): 3
18/18 - 0s - loss: 0.0323 - accuracy: 0.9930
12/12 - 0s - loss: 1.9742 - accuracy: 0.7995
confusion_matrix(C): tf.Tensor(
[[379   4]
 [  0 191]], shape=(2, 2), dtype=int32)
```

학습률 0.01에 epoches를 400으로 돌렸을 때 약 99%의 정확도를 보였습니다.

epochs=400과 learning_rate = 0.01 기준으로 손실 함수에 따른 차이

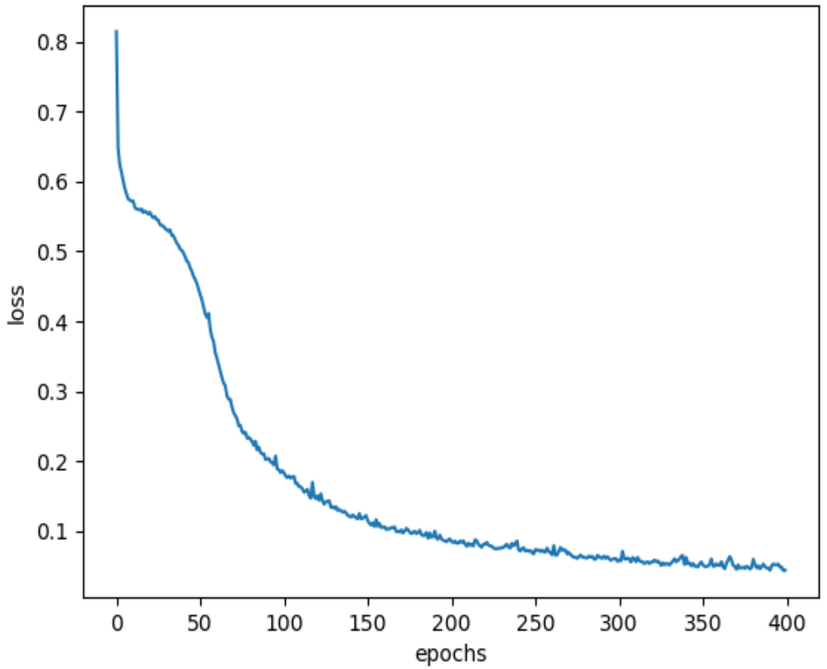
1. MSE




```
len(model.layers): 3
18/18 - 0s - loss: 0.0333 - accuracy: 0.9686
12/12 - 0s - loss: 0.1715 - accuracy: 0.8047
confusion_matrix(C): tf.Tensor(
[[178  17]
 [  1 378]], shape=(2, 2), dtype=int32)
```

MSE로 돌렸을 때 약 98%의 정확도를 보였습니다.

2. categorical_crossentropy




```
len(model.layers): 3
18/18 - 0s - loss: 0.0412 - accuracy: 0.9878
12/12 - 0s - loss: 1.0015 - accuracy: 0.8177
confusion_matrix(C): tf.Tensor(
[[366   7]
 [  0 201]], shape=(2, 2), dtype=int32)
>>>
```

categorical_crossentropy로 했을 때 약 99%의 정확도를 보였습니다.

학습 결과 분석

keras 는 Adam, learning_rate 는 0.01, epoches 는 400 번 , 손 실 함 수 는 categorical_crossentropy로 하는 것이 가장 효율적이고 높은 정확도를 얻게 되는 것인 거 같습니다.

learning_rate와 epoches에서는 learning_rate가 낮을수록 epoches는 높게 돌려야 좀 더 섬세하고 높은 정확도가 나올 것이나 컴퓨터에 따라 낮은 learning_rate와 높은 epoches를 힘들어하는 컴퓨터도 있을 것입니다.

오히려 너무 낮은 learning_rate와 높은 epoches로 돌렸을 때 정확도에 그만큼의 변화가 없었고 100% 정확도를 보이고 있는 상태에서는 오히려 떨어지거나 쓸떼없이 학습하고 있는 모습을 보였습니다.

적절한 learning_rate와 epoches를 찾고 설정하는 것이 중요합니다.