Machine Learning Report

04

학번: 2017112200

이름: 신현호

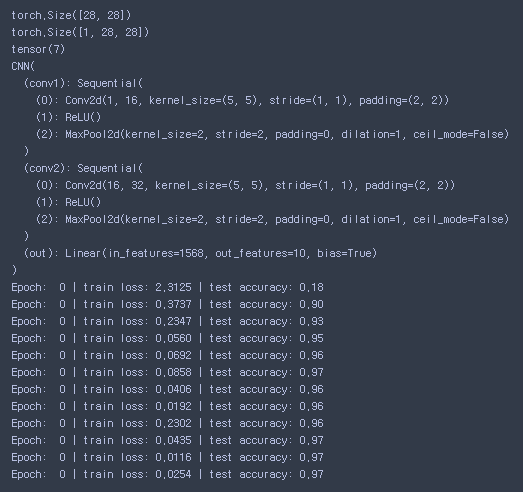
1. CNN & MNIST
2. Change the current kernel size of the program to different size. (Change ‘kernel\_size’ parameter of ‘Conv2D’ function.) Repeat this three times and compare the results.

<Program Code>

5일 때



<Result>

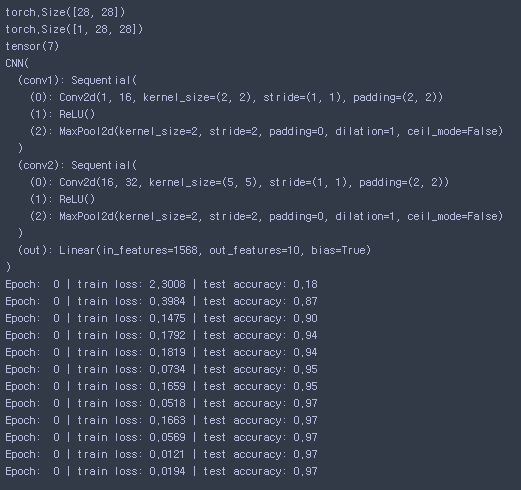


2일 때

<Program Code>



<Result>

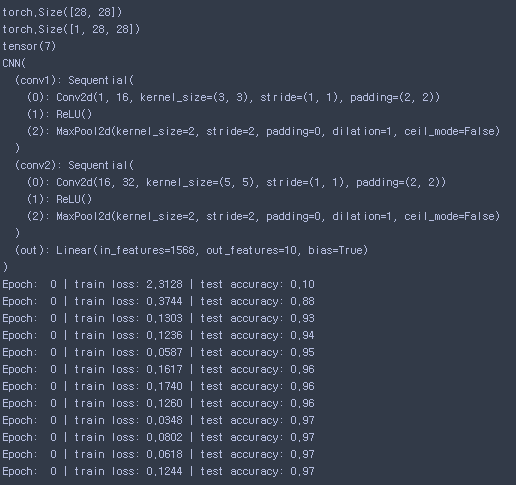


3일 때

<Program Code>



<Result>



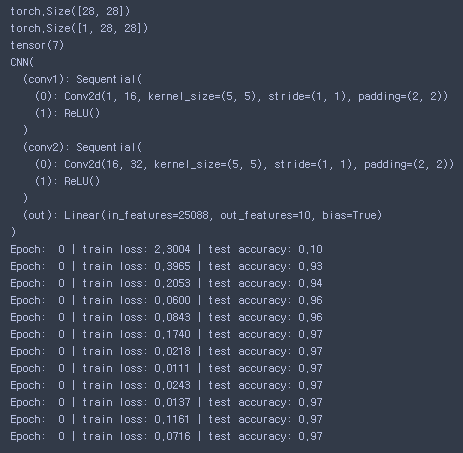
* 세 경우 대체적으로 값이 비슷했지만, train loss가 가장 큰 경우는 kernel\_size가 3일 때였고, 이 때 모든 경우를 통틀어, 가장 최소의 값이 나왔다.

1. remove pooling layer in the program (you can remove ‘MaxPool2D’ function) and compare the results.

<Program Code>



<Result>



* pooling layer가 있을 때와 없을 때, 가장 높은 정확도는 0.97인데, 0.97의 빈도가 MaxPool2d 함수를 없앴을 때, 더 늘어났다.

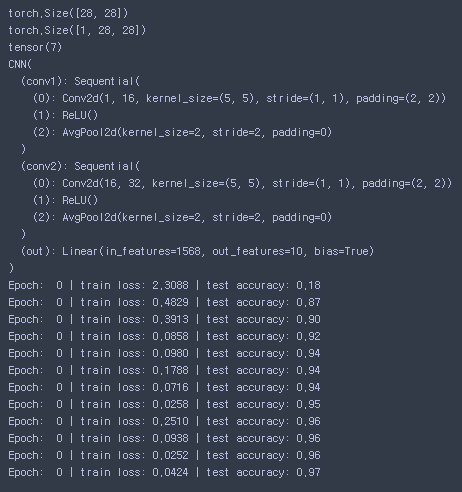
1. Change pooling layer in the program(ex. AvgPool2d, AdoptiveAvgPool2d, etc.) repeat this three times and compare the results.

<Program Code>

AvgPool2d



<Result>

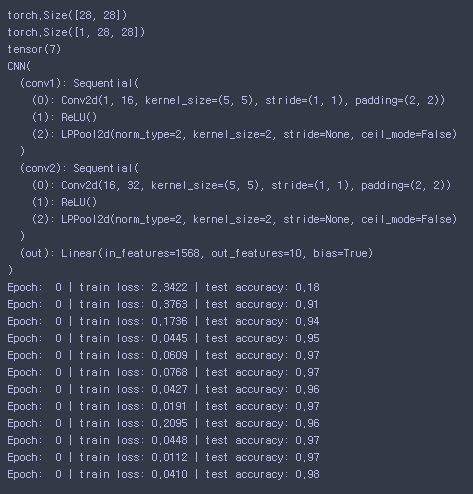


<Program Code>

LPPool2d



<Result>

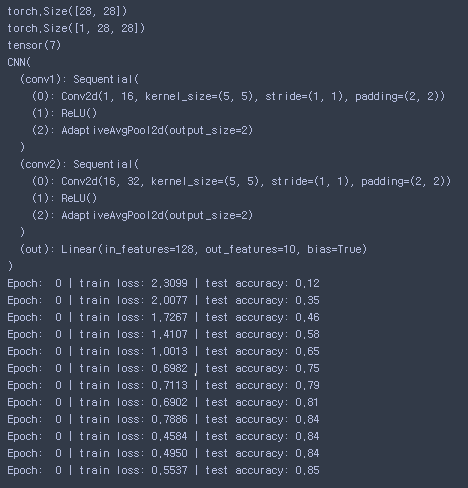


<Program Code>

AdaptiveAvgPool2d



<Result>



* 정확도면에서 LPPool2d, AvgPool2d, AdaptivePool2d 순으로 성능이 나은 것을 확인할 수 있다. 특히, AdaptiveAvgPool2d는 다른 함수들에 비해 현저히 낮은 성능을 보였다.

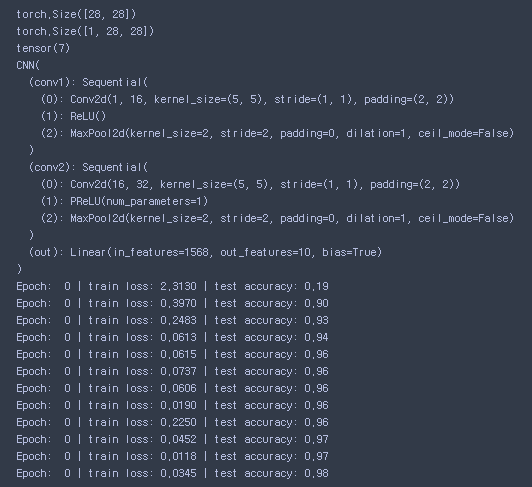
1. change the current activation function to other non-linear activation function (e.g. sigmoid, tanh, etc). You can do so by nn.Sigmoid() to nn.ReLU(), nn.Tanh(), etc. Repeat this five times and compare the results.

<Program Code>

PReLU



<Result>

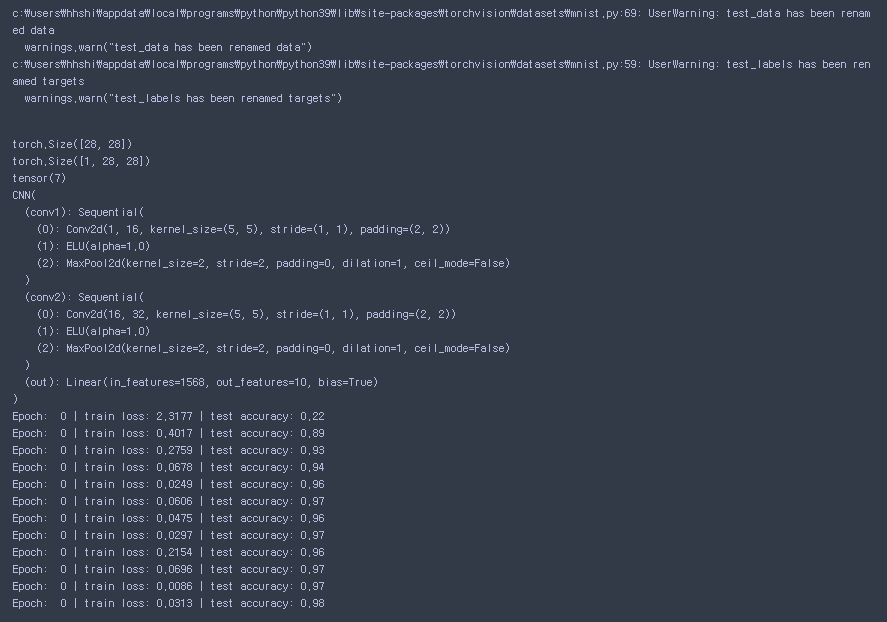


<Program Code>

ELU



<Result>

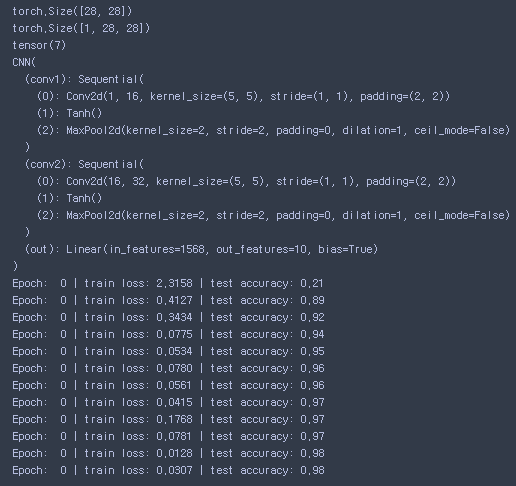


<Program Code>

Tanh



<Result>

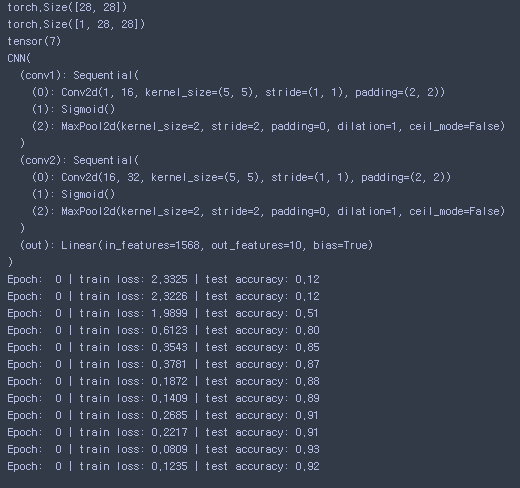


<Program Code>

Sigmoid



<Result>

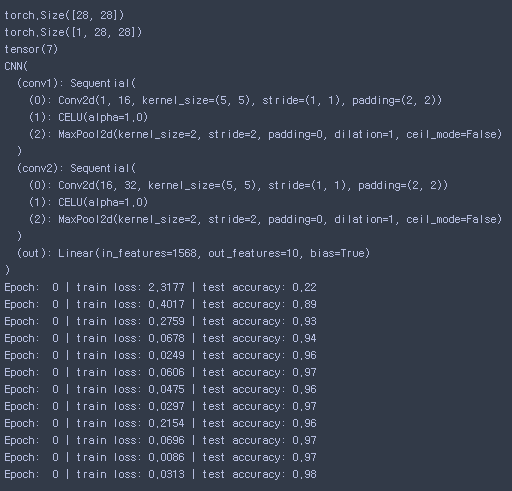


<Program Code>

CELU



<Result>



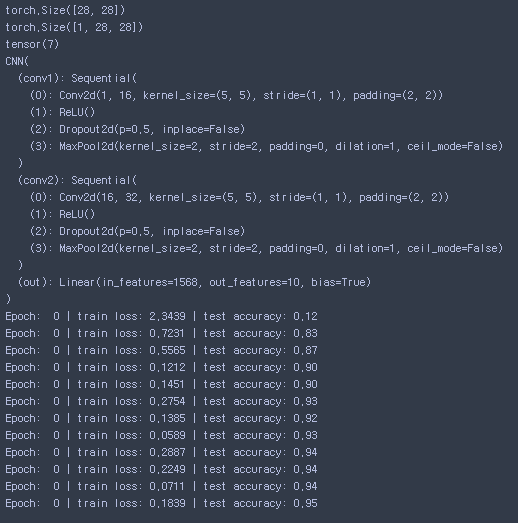
* 다른 함수들을 썼을 때와는 달리, sigmoid 함수를 썼을 때는 성능이 나쁘진 않지만 비교적으로 성능이 다른 함수들에 비해 낮음을 확인할 수 있었다.

1. Add dropout layers to the hidden layer and compare the results.

<Program Code>



<Result>



- 각 계층에 dropout 함수를 추가하면, 정확도가 낮은 편은 아니지만, 없을 때보단 낮았음을 확인할 수 있었다.

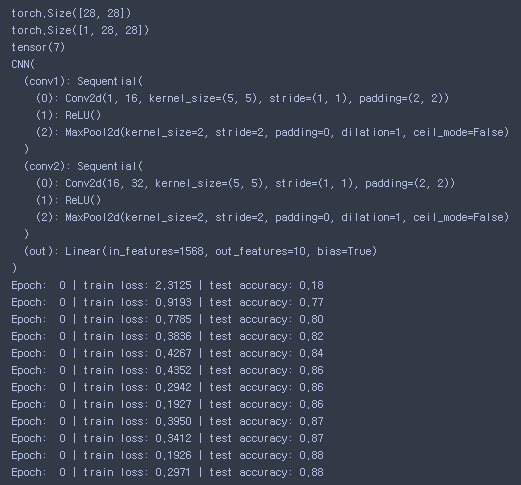
1. change the current optimization method to other optimization methods (e.g. adam, adaGrad, RMSProp, adaDelta, etc). You can use torch.optim.Adam, etc. Repeat this three times and compare the results.

<Program Code>

Adagrad

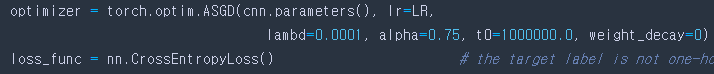


<Result>

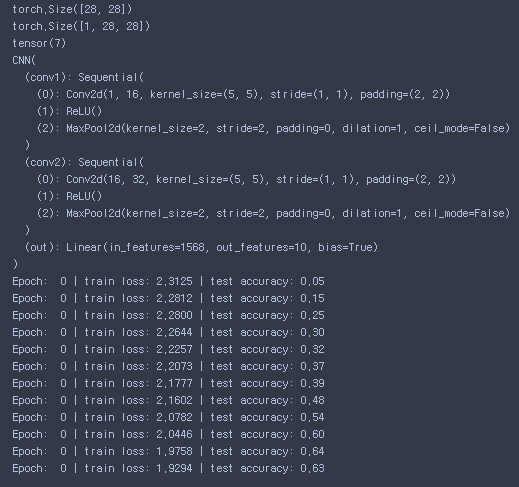


<Program Code>

ASGD

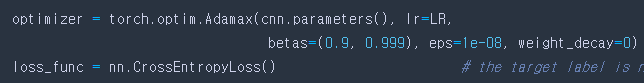


<Result>

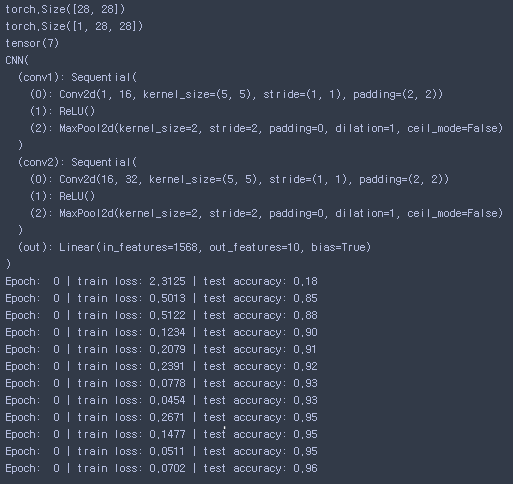


<Program Code>

Adamax



<Result>



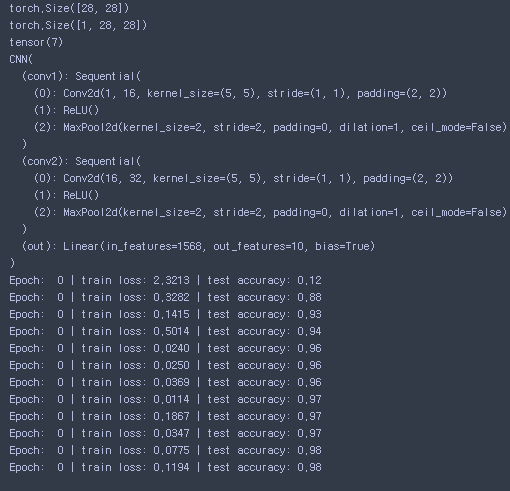
* Adamax, Adagrad, ASGD 순으로 정확도가 나열되었다. ASGD는 Adamax에 비하면 현저히 낮은 정확도를 보였다.

1. now add the Xavier weight initialization method and compare the results. (use torch.nn.init.xavier\_uniform)

<Program Code>



<Result>

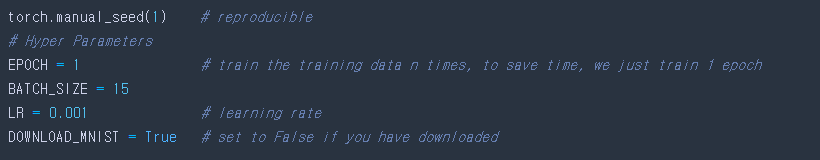


* 각 특성에 맞게 초기화를 진행함으로써 정확도가 전체적으로 올라갔다.

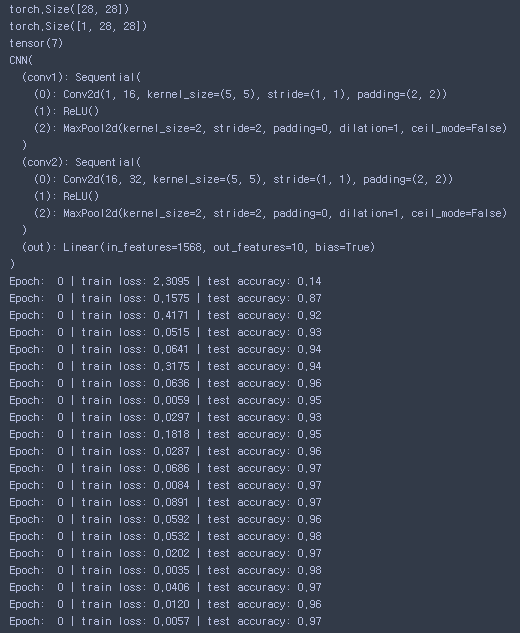
1. choose ONE other parameters of CNN program (e.g. number of hidden nodes, epochs, batch normalization, etc). Change the value of this parameter and compare the results.

<Program Code>

BATCH\_SIZE를 15로 바꿈



<Result>



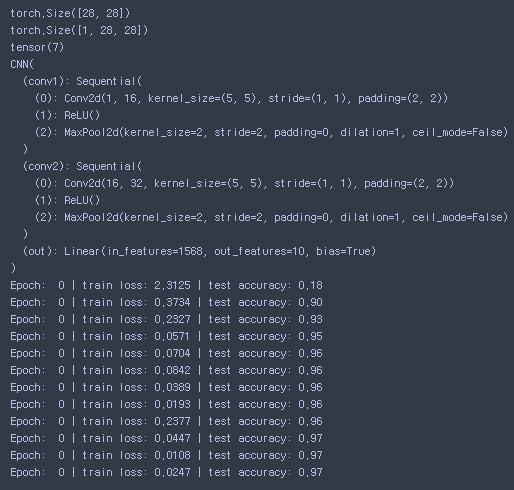
* BATCH\_SIZE를 줄이니, 원래는 없었던 0.98의 정확도가 생겼다. 계산해야할 양이 줄다보니, 보다 높은 정확도도 생겼다.

1. (\*optional\*) choose Adam optimization method and use L2 (ridge) regularization method this time. You can do so by setting ‘weight\_decay’ value in optimization method (torch.optim.Adam) to a tiny number (e.g. 1e-5). Compare the results of using regularization.

<Program Code>



<Result>

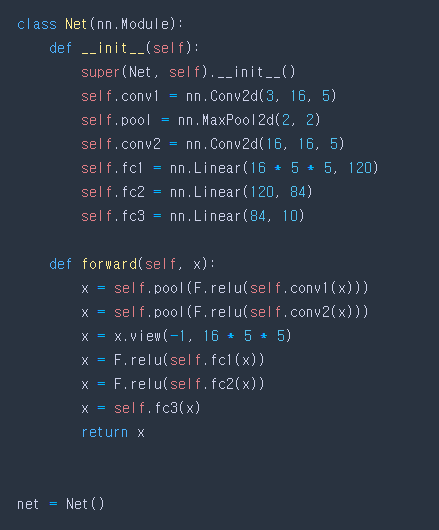


* L2 정규화는 모든 파라미터들을 남겨놓고, 각 값을 최소화한다. Weight\_decay의 값이 작아서 그런지, 별 다른 차이를 보이지 않았다.

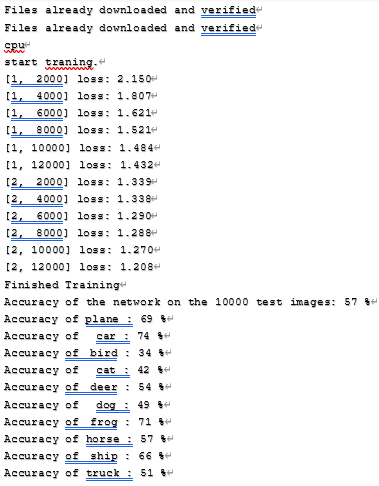
1. CNN & CIFAR-10
2. change the current kernel size of the program to different size. (Change ‘kernel\_size’ parameter of ‘Conv2D’ function.) Repeat this three times and compare the results

<Program Code>

kernel\_size = 5

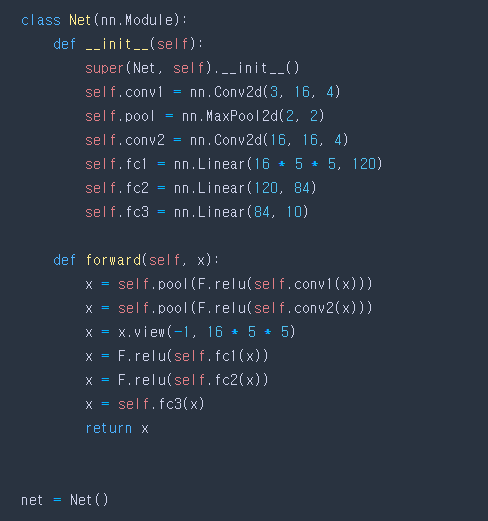


<Result>

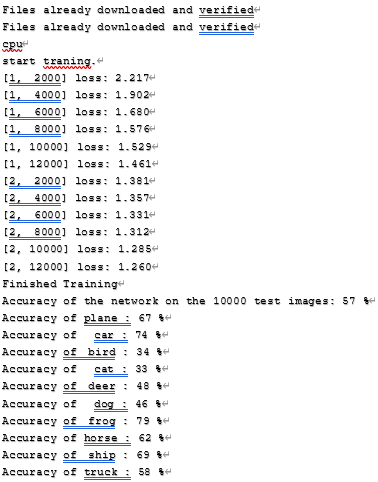


<Program Code>

kernel\_size = 4



<Result>

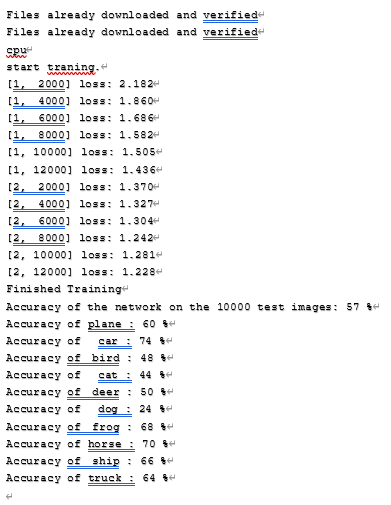


<Program Code>

kernel\_size = 2(conv2는 5로 하지 않으면 계속 오류가 나서 5로 놓고 진행했습니다.)



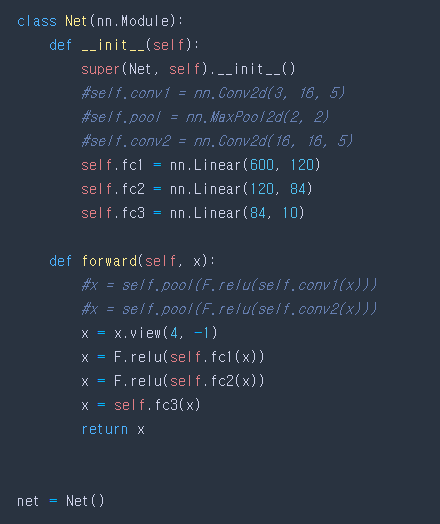
<Result>



* 각 그림마다의 정확도 차이는 있었지만, 총 정확도는 모두 같았다.

1. remove pooling layer in the program (you can remove ‘MaxPool2D’ function) and compare the results.

<Program Code>



<Result>

오류가 났습니다.

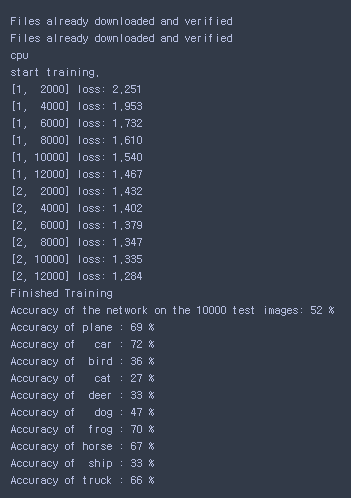
1. change pooling layer in the program (ex. AvgPool2d, AdoptiveAvgPool2d, etc.) repeat this three times and compare the results

<Program Code>

AvgPool2d

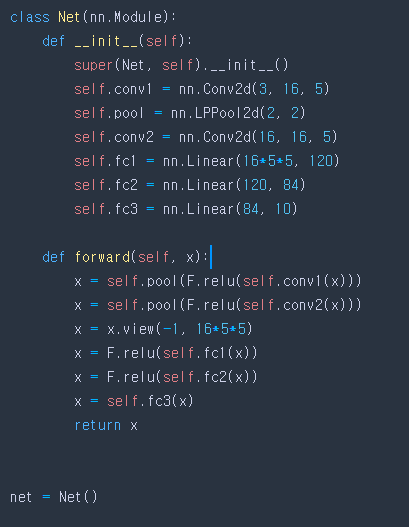


<Result>

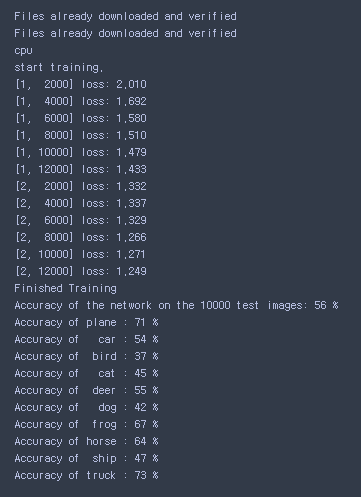


<Program Code>

LPPool2d



<Result>

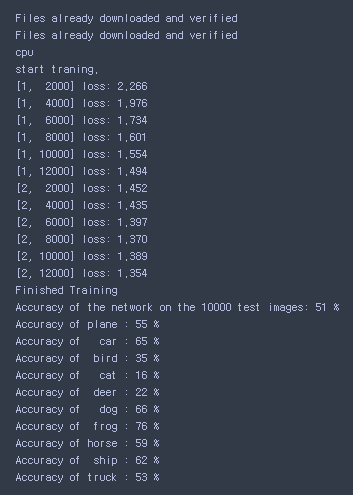


<Program Code>

AdaptiveAvgPool2d



<Result>



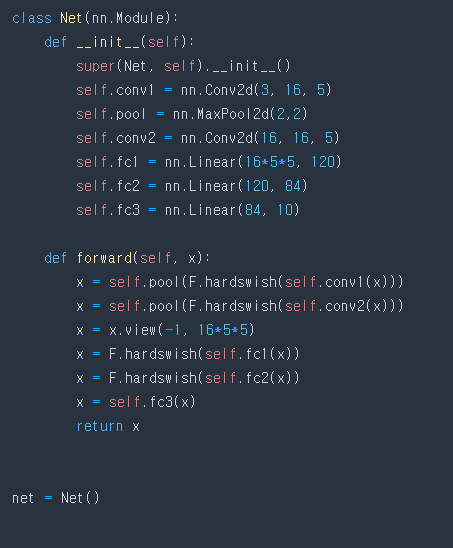
* LPPool2d, AvgPool2d, AdpativeAvgPool2d 순으로 총 정확도의 순서가 매겨진다.

LPPool2d는 새의, 나머지 두 함수는 고양이의 이미지를 학습할 때 가장 낮은 정확도를 보였다.

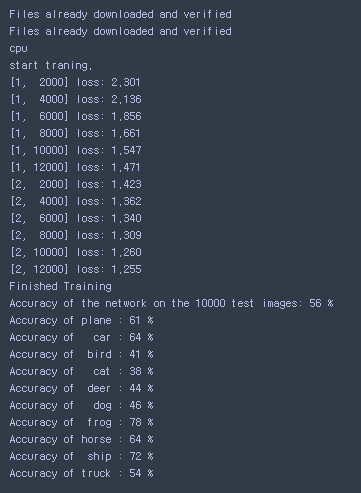
1. change the current activation function to other non-linear activation function (e.g. sigmoid, tanh, etc). You can do so by nn.Sigmoid() to nn.ReLU(), nn.Tanh(), etc. Repeat this five times and compare the results.

<Program Code>

hardswish



<Result>

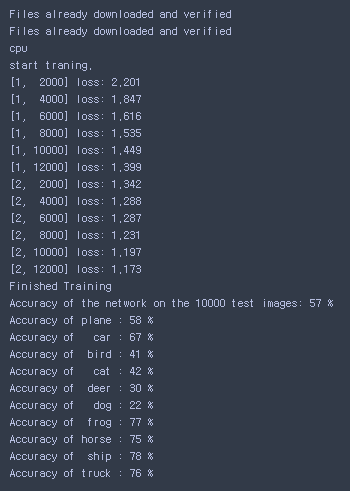


<Program Code>

relu6

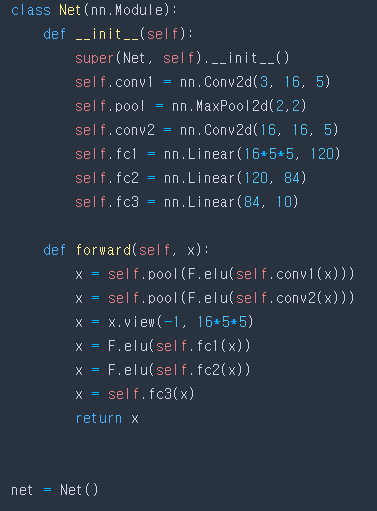


<Result>

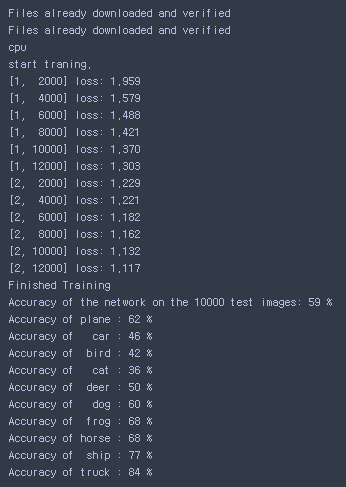


<Program Code>

elu

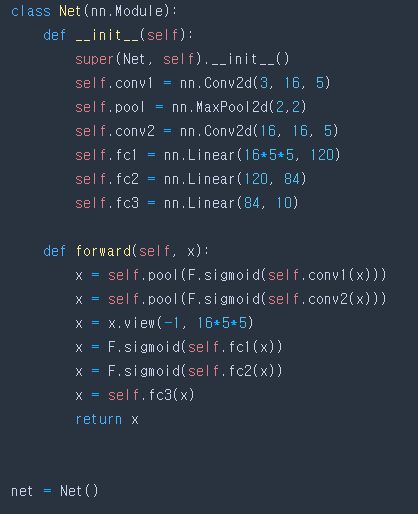


<Result>

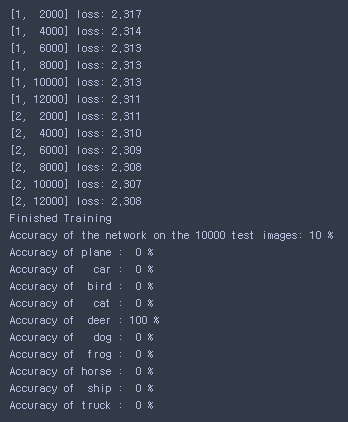


<Program Code>

sigmoid



<Result>

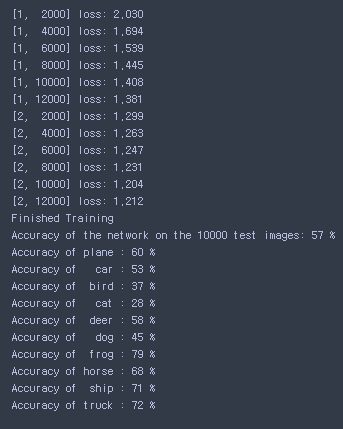


<Program Code>

tanh



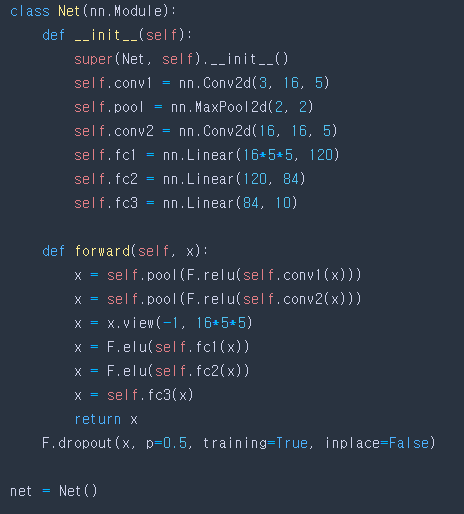
<Result>



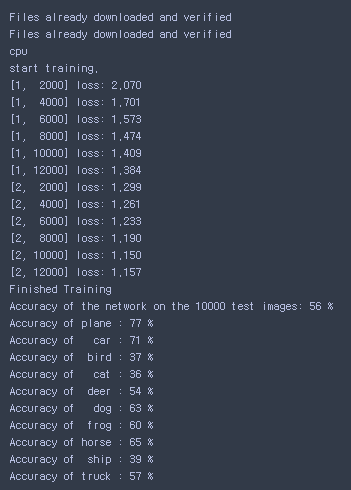
* sigmoid 함수를 제외하고 다른 함수들은 모두 정확도가 대체적으로 50 중후반대가 나왔다. Sigmoid를 썼을 때는 loss도 현저히 높았고, 각 이미지에 대한 정확도는 사슴의 경우만 100프로가 나오고, 나머지는 0프로가 나왔었다.

1. add dropout layers to the hidden layer and compare the results.

<Program Code>



<Result>



* 전체적인 정확도는 비슷했으나, 각 이미지의 정확도를 보면, 배 이미지에선 편차가 심했다.

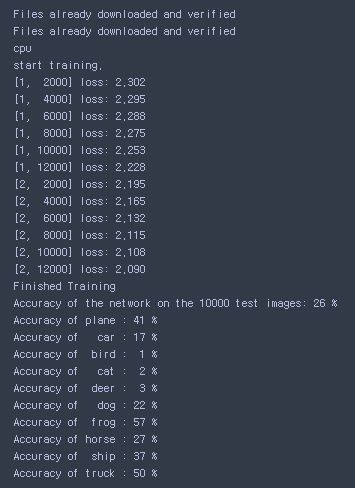
1. change the current optimization method to other optimization methods (e.g. adam, adaGrad, RMSProp, adaDelta, etc). You can use torch.optim.Adam, etc. Repeat this three times and compare the results.

<Program Code>

Adadelta

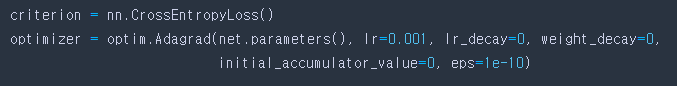


<Result>

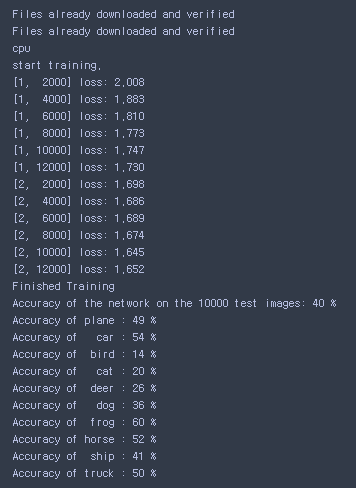


<Program Code>

Adagrad

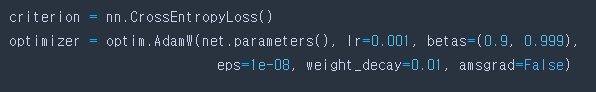


<Result>



<Program Code>

Adamw



<Result>



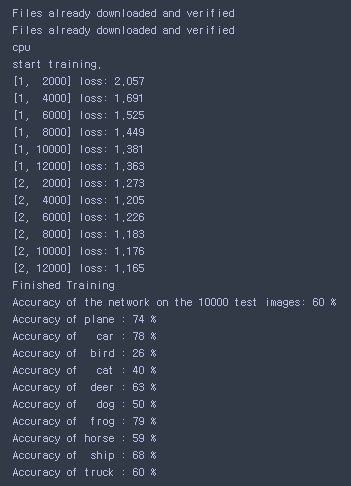
* Adamw, Adagrad, Adadelta 순으로 정확도가 나열이 되었고, Adadelta의 경우에는 현저히 정확도가 낮았다. 특히, 한 자리 수의 정확도인 이미지들도 있었다.

1. now add the Xavier weight initialization method and compare the results. (use torch.nn.init.xavier\_uniform)

<Program Code>



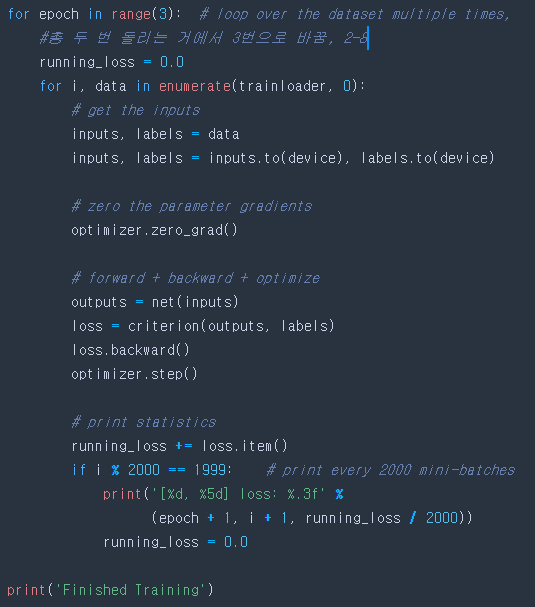
<Result>



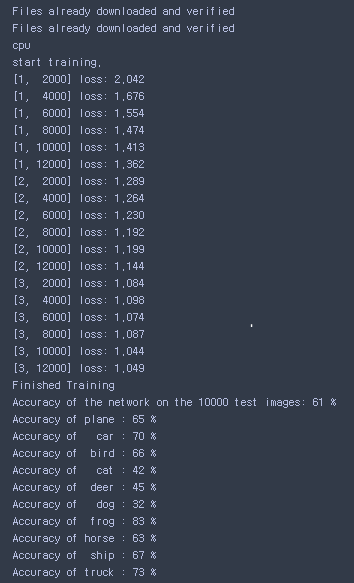
* 전체적인 정확도가 올랐다.

1. choose ONE other parameters of CNN program (e.g. number of hidden nodes, epochs, batch normalization, etc). Change the value of this parameter and compare the results.

<Program Code>



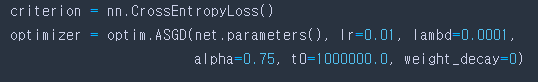
<Result>



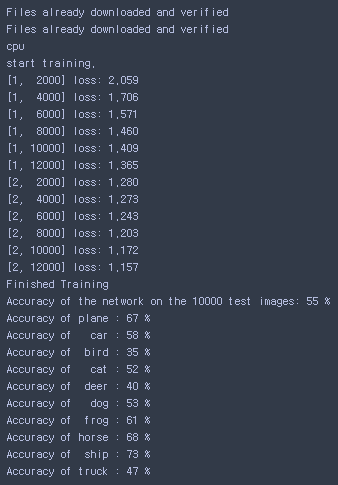
* 전체적인 정확도가 올랐으며, 횟수를 더 많이 하여 학습을 시킨만큼, loss의 최종값도 1.2에서 1.0대로 줄어들었다.

1. (\*optional\*) choose Adam optimization method and use L2 (ridge) regularization method this time. You can do so by setting ‘weight\_decay’ value in optimization method (torch.optim.Adam) to a tiny number (e.g. 1e-5). Compare the results of using regularization.

<Program Code>



<Result>

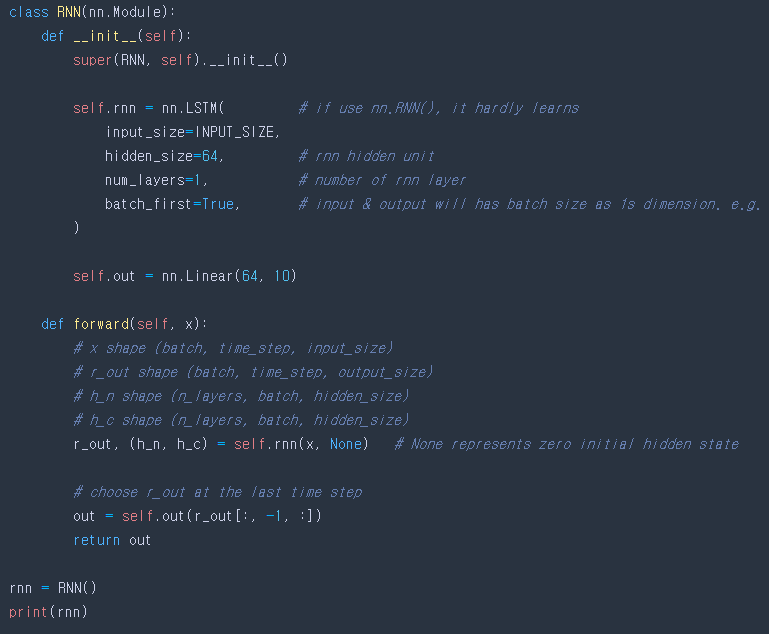


* 비슷하긴하나, 전체적인 정확도는 약간 내려갔다. 여전히 bird 이미지에서 가장 낮은 정확도를 보였다.

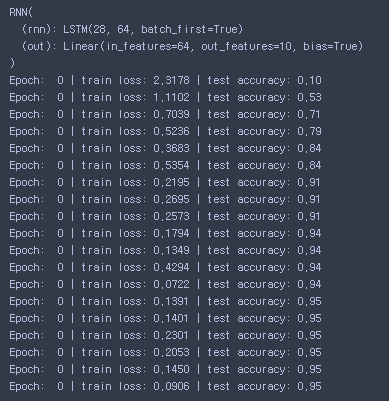
1. RNN & MNIST
2. refer to the RNN\_Mnist program in e-class
3. change the number of hidden nodes in the program three times and compare the results.

<Program Code>

kernel\_size = 64

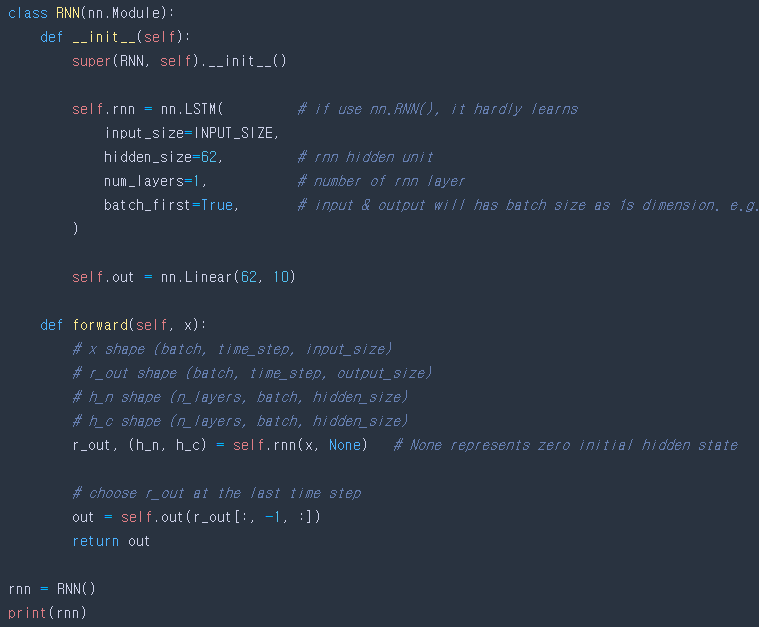


<Result>

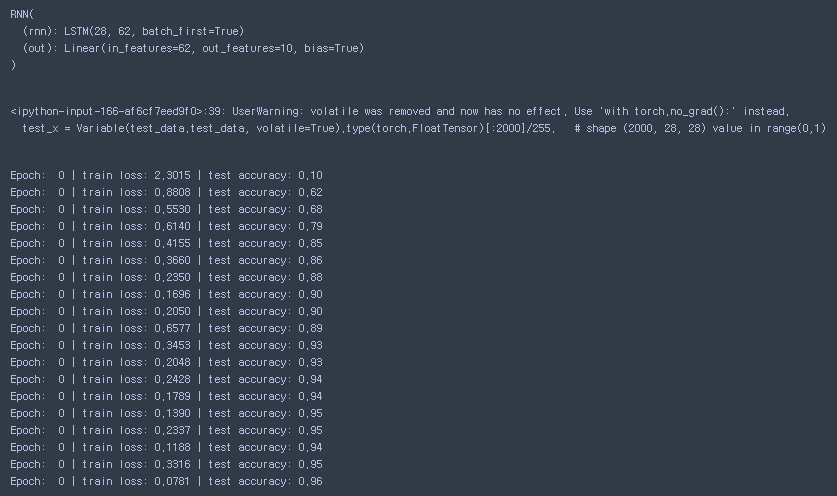


<Program Code>

kernel\_size = 62

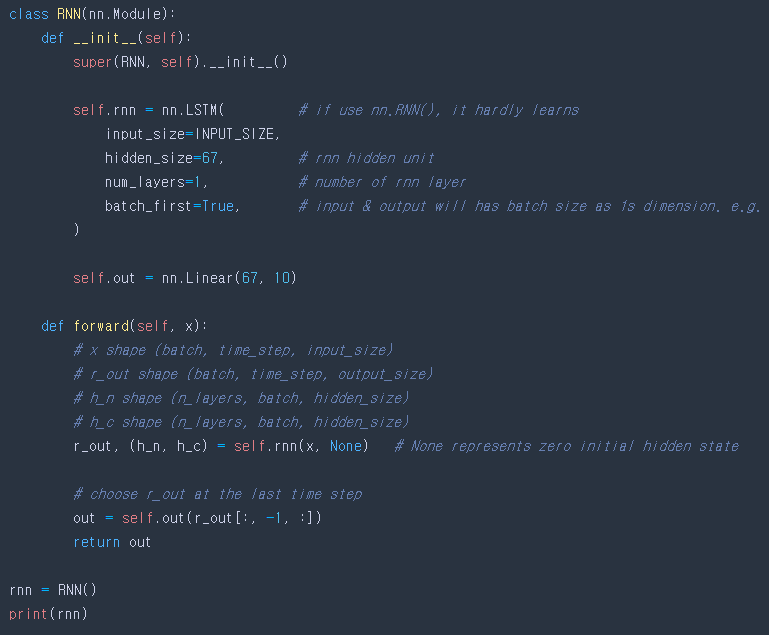


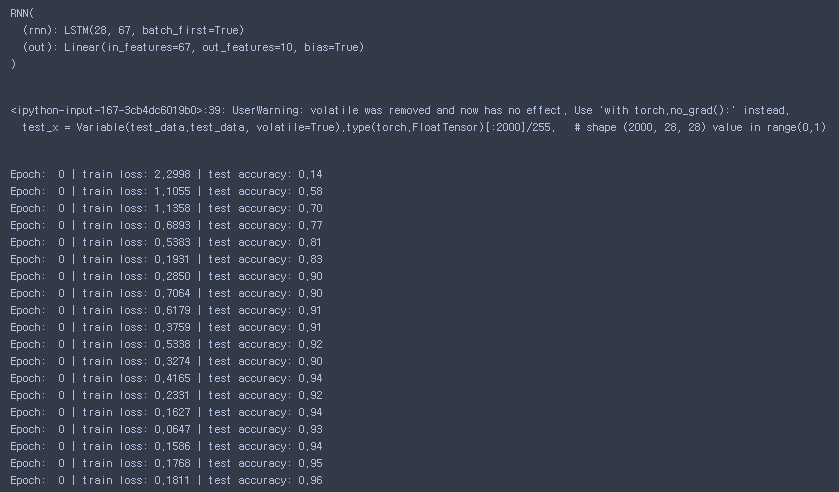
<Result>



<Program Code>

kernel\_size = 67



<Result>  


* kernel\_size를 65, 62, 67로 놓고 실행 시킨 결과, 각 결과끼리의 차이가 크지 않았다.

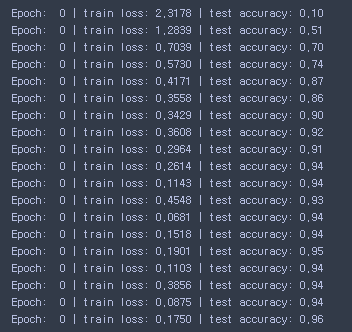
1. change the current optimization method to other optimization methods (e.g. adam, adaGrad, RMSProp, adaDelta, etc). Repeat this three times and compare the results.

<Program Code>

AdamW



<Result>



<Program Code>

Adamax



<Result>

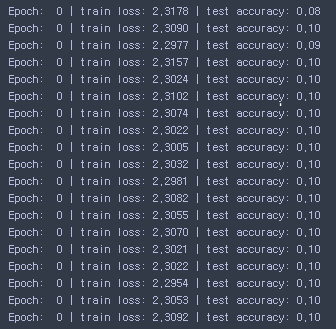


<Program Code>

ASGD



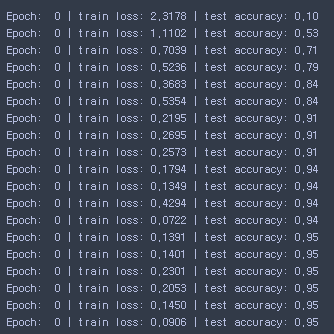
<Result>



* AdamW 함수를 썼을 때가 Adamax를 쓸 때보다 조금 나았고, ASGD를 썼을 때는 정확도가 현저히 낮았고, train loss도 매우 컸다.

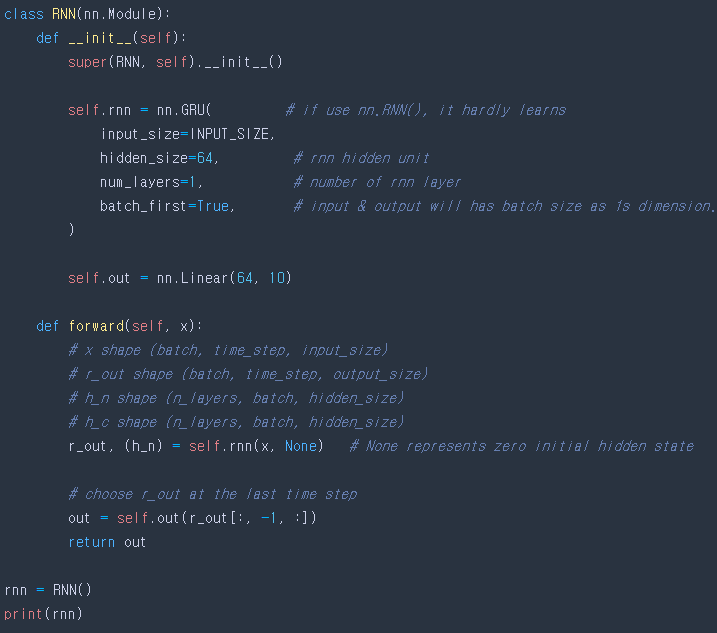
1. change LSTM to GRU (or vice versa). Compare the results.

<LSTM 사용했을 때의 결과>

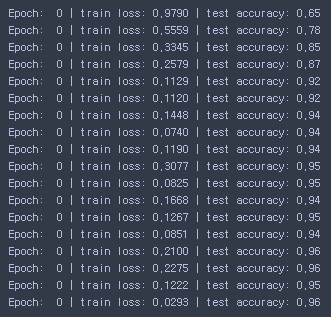


<Program Code>

GRU 사용 시



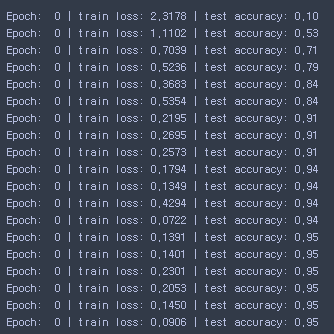
<Result>



* LSTM을 사용했을 때는, 처음 두 시도 횟수의 train loss가 각각 2와 1을 넘었지만, GRU를 사용했을 때는 train loss가 항상 1보다 낮았다. 정확도는 둘이 엇비슷했다.

1. choose ONE other parameters of RNN program (e.g. batch\_size, epochs, etc). Change the value of this parameter and compare the results.

<EPOCH = 1 일 때>

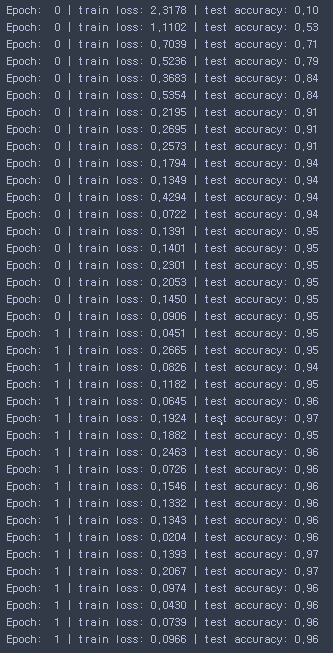


<Program Code>

EPOCH = 2일 때



<Result>



* EPOCH를 2로 늘렸을 때의 결과는 EPOCH가 원래 값인 1일 때의 결과값들을 포함한 결과가 나온다. 결과창에서, Epoch: 1일 때의 결과가 Epoch가 0일 때의 결과보다 좋다.

1. compare the accuracy of RNN for Mnist with that of CNN.

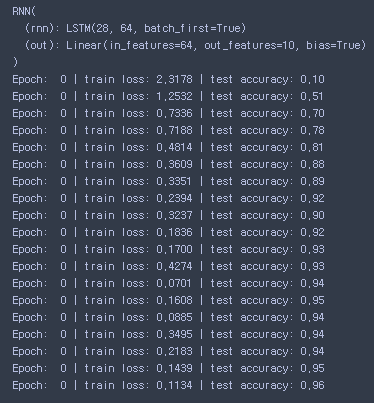
* 결과로 봤을 때는, CNN이 더 높은 정확도를 보였다.

1. (\*optional\*) choose Adam optimization method and use L2 (ridge) regularization method this time. You can do so by setting ‘weight\_decay’ value in optimization method to a tiny number (e.g. 1e-5). Compare the results of using regularization.

<Program Code>



<Result>



* 가장 작은 train loss 값을 갖는 경우는 regularization을 하지 않았을 때였고, 두 경우 다 비슷한 정확도를 보인다.

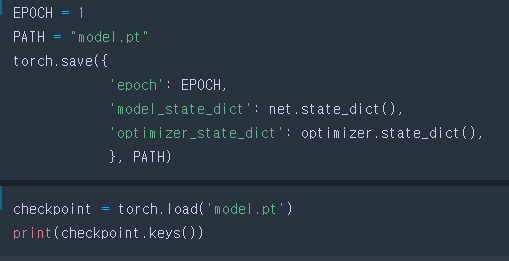
1. (\*optional\*) use dropout technique and compare the results.

<Program Code>

<Result>

1. save your checkpoint (save STATE\_DICT of your model. your checkpoint must contain EPOCH and STATE\_DICT of the model and optimizer.) and then, load the saved checkpoint and make sure it works. (don’t save your all model. you have to save your parameters only.)

<Program Code>



<Result>

