

Project 3 - Fashion-MNIST Classification

1. Environment

```
import platform
import os
from platform import python_version
print('os name:',platform.system(),platform.release())
print('python version :',python_version())
print('torch version :',torch.__version__)
print('torchvision version :',torchvision.__version__)
print('matplotlib version :',matplotlib.__version__)
print('numpy version :',numpy.__version__)
```

```
os name: Linux 5.4.109+
python version : 3.7.10
torch version : 1.8.1+cu101
torchvision version : 0.9.1+cu101
matplotlib version : 3.2.2
numpy version : 1.19.5
```

- OS : Linux 5.4.109
- python version : 3.7.10
- torch version : 1.8.1
- torchvision version: 0.9.1
- matplotlib version : 3.2.2
- numpy version : 1.19.5

2. Network architecture description

2-1 Fully Connected Network

```
class FC_Network(nn.Module):
    def __init__(self):
        super(FC_Network, self).__init__()
        self.fc1=nn.Linear([28*28,512,bias=True])
        self.fc2=nn.Linear(512,10,bias=True)
        self.relu=nn.ReLU()
        torch.nn.init.xavier_uniform_(self.fc1.weight)
        torch.nn.init.xavier_uniform_(self.fc2.weight)

    def forward(self,x):
        x=self.fc1(x.reshape(x.shape[0],-1))
        x=self.relu(x)
        x=self.fc2(x)
        return x
```

Fully connected network 은 nn.Module를 상속받아 생성했고, 구성은 차례로

- input size=784, output size=512 인 fully connected layer
- ReLU activation function
- input size=512, output size=10 인 fully connected layer

로 구성되어 있습니다.

Input data가 들어오게 되면, [batchsize,1,28,28]의 3차원의 형태로 각각의 이미지가 들어오기 때문에 fully connected layer를 통과하기 위해서 reshape을 통해서 [batchsize,784]의 1차원 형태로 각각의 이미지를 변환되게 됩니다.

그 후에 fully connected layer에 의해 [batchsize,512]의 형태로 변환되게 되고, ReLU activation function을 통과하게 됩니다.

마지막으로 fully connected layer에 의해 [batchsize,10]의 형태로 각각의 이미지에 대해 크기가 10인 vector의 형태로 output data가 생성됩니다.

2-2 LeNet-5

```
class Lenet_5(nn.Module):
    def __init__(self, softmax_exist=False,
                  activation_function='tanh',
                  first_channel=6,
                  second_channel=16,
                  third_channel=120,
                  use_batchnorm=False,
                  pooling_layer='avg',
                  use_dropout=False):
        super(Lenet_5, self).__init__()
        self.con1=nn.Conv2d(in_channels=1, out_channels=first_channel, kernel_size=5, stride=1, padding=2)
        self.con2=nn.Conv2d(in_channels=first_channel, out_channels=second_channel, kernel_size=5, stride=1)
        self.con3=nn.Conv2d(in_channels=second_channel, out_channels=third_channel, kernel_size=5, stride=1)

        self.use_batchnormflag=use_batchnorm
        self.batchnorm1=nn.BatchNorm2d(first_channel)
        self.batchnorm2=nn.BatchNorm2d(second_channel)
        self.batchnorm3=nn.BatchNorm2d(third_channel)

        if activation_function=='tanh':
            self.activation_function=nn.Tanh()
        elif activation_function=='relu':
            self.activation_function=nn.ReLU()
        elif activation_function=='leaky_relu':
            self.activation_function=nn.LeakyReLU()

        if pooling_layer=='avg':
            self.pooling_layer=nn.AvgPool2d(2)
        elif pooling_layer=='max':
            self.pooling_layer=nn.MaxPool2d(2)

        self.fc1=nn.Linear(third_channel, 84)
        self.fc2=nn.Linear(84, 10)

        self.sflag=softmax_exist
        self.softmax=nn.Softmax(1)

        self.dflag=use_dropout
        self.dropout=nn.Dropout(0.3)

        torch.nn.init.xavier_uniform_(self.fc1.weight)
        torch.nn.init.xavier_uniform_(self.fc2.weight)
```

```

def forward(self, x):
    x=self.con1(x)
    if self.use_batchnormflag==True:
        x=self.batchnorm1(x)
    x=self.activation_function(x)
    x=self.pooling_layer(x)

    x=self.con2(x)
    if self.use_batchnormflag==True:
        x=self.batchnorm2(x)
    x=self.activation_function(x)
    x=self.pooling_layer(x)

    x=self.con3(x)
    if self.use_batchnormflag==True:
        x=self.batchnorm3(x)
    x=self.activation_function(x)

    x=x.reshape(x.shape[0],-1)

    x=self.fc1(x)
    if self.dflag:
        x=self.dropout(x)
    x=self.activation_function(x)
    x=self.fc2(x)
    if self.sflag:
        x=self.softmax(x)
    return x

```

LeNet-5은 전체적으로 3개의 convolutional layer와 2개의 fully connected layer로 구성되어 있고, nn.Module를 상속 받아서 생성했습니다. 구성은 차례로

- input channel=1, output channel=6, kernel size=5, stride=1, padding=2인 convolutional layer
- tanh activation function
- average pooling layer
- input channel=6, output channel=16, kernel size=5, stride=1, padding=0인 convolutional layer
- tanh activation function
- average pooling layer
- input channel=16, output channel=120, kernel size=5, stride=1, padding=0인 convolutional layer
- tanh activation function
- input size=120, output size=84 인 fully connected layer
- tanh activation function
- input size=84, output size=10 인 fully connected layer

로 구성되어 있습니다.

input data [batchsize,1,28,28] 가 들어오게 되면 LeNet-5 구조에서는 [1,32,32] 의 형태로 이미지가 들어와야 되기 때문에 첫번째 convolutional layer에서 padding을 2를 줘야 됩니다.

첫번째 convolutional layer를 통과하게 되면, [batchsize,6,28,28] 로 형태가 변하게 되고, tanh activation function과 크기 2짜리 average pooling layer를 통과하게 되면 [batchsize,6,14,14] 가 됩니다.

그후에 2번째 convolutional layer, tanh activation function, 크기 2짜리 average pooling layer를 통과하게 되면 [batchsize,16,5,5]의 형태가 변하게 됩니다.

마지막 convolution layer과 tanh activation function을 통과하게 되면 [batchsize,120,1,1]로 형태가 됩니다.

그후에 fully connected layer를 통과하기 위해서 이미지가 1차원의 형태로 변환하게 되서 [batchsize,120]로 바뀝니다.

첫번째 fully connected layer, tanh activation function을 통과하게 되면 [batchsize,84]로 형태가 변하게 되고

마지막 fully connected layer를 통과하게 되면서 최종적으로 [batchsize, 10]의 형태로 각각의 이미지에 대해서 크기가 10인 vector의 형태로 output data가 생성되게 됩니다.

모델을 정의하는 부분에 있어서 LeNet-5의 학습율을 높이기 위해서 변수들을 받아서 모델을 설정할 수 있게 했습니다.

- softmax : True, False

MSELoss를 loss function으로 사용하는 경우, model의 마지막 layer에 softmax layer를 추가해야 되어 설정값으로 추가할 수 있도록 했습니다. True로 설정되어 있는 경우, 모델의 마지막 layer에 softmax layer를 추가하게 됩니다.

- activation_function : 'tanh', 'relu', 'leaky_relu'

activation function을 따로 지정할 수 있고, 기본값으로는 tanh 함수를 사용하고 있고, 설정에 따라서 tanh, relu, leaky_relu를 activation 함수로 사용할 수 있습니다.

- first channel, second channel, third channel : int

first channel, second channel, third channel은 각각 첫번째, 두번째, 세번째 convolutional layer에서 out channel에 해당하는 채널의 수이며, 설정에 따라 조정할 수 있게 했습니다.

- use_batchnorm : True, False

use_batchnorm이 설정되어 있는 경우, convolutional layer를 통과한 이후 batch normalization을 진행하도록 설정했습니다.

- pooling_layer : 'avg', 'max'

pooling layer도 종류도 설정할 수 있게 하였습니다. average pooling layer 또는 max pooling layer를 사용할 수 있습니다.

- use_dropout : True, False

use_dropout이 설정되어 있는 경우, 첫번째 fully connected layer후에 dropout을 적용하도록 하였습니다.

3. loss function and optimizer

3-1 Fully Connected Layer - optimizer, loss function function

```

def train_FC(training_epoch, batch_size, learning_rate):
    device='cpu'
    if torch.cuda.is_available():
        device='cuda'
    print('using device :', device)

    train_data_loader, test_data_loader=get_loader(batch_size=batch_size)

    model=FC_Network().to(device)
    criterion=nn.CrossEntropyLoss().to(device)
    optimizer=torch.optim.Adam(model.parameters(), lr=learning_rate)
    print('hyper parameter')
    print('training epoch :', training_epoch)
    print('batch_size :', batch_size)
    print('learning rate :', learning_rate)

    epoch_num=0
    loss_per_epoch=[]
    batch_len=len(train_data_loader)
    for epoch in range(training_epoch):
        epoch_num+=1
        training_loss=0
        for imgs, labels in train_data_loader:
            imgs=imgs.to(device)
            labels=labels.to(device)
            predictions=model(imgs)
            loss_function=criterion(predictions, labels)
            optimizer.zero_grad()
            loss_function.backward()
            optimizer.step()
            training_loss+=loss_function.item()/batch_len
        print('epoch :', epoch_num, '##training_loss :', training_loss)
        loss_per_epoch.append(training_loss)

```

다음은 Fully Connected Layer를 학습시킬 때 사용한 코드이며, loss function으로는 CrossEntropyLoss를 사용했으며, optimizer로는 Adam optimizer를 사용했습니다.

CrossEntropyLoss는 LogSoftmax 와 NLLLoss를 순차적으로 진행해서 나오는 loss 이며, 첫번째로

LOGSOFTMAX

CLASS torch.nn.LogSoftmax(*dim=None*)

[\[SOURCE\]](#)

Applies the $\log(\text{Softmax}(x))$ function to an n-dimensional input Tensor. The LogSoftmax formulation can be simplified as:

$$\text{LogSoftmax}(x_i) = \log \left(\frac{\exp(x_i)}{\sum_j \exp(x_j)} \right)$$

Logsoftmax 는 tensor의 각 element들의 지수승을 취하고, 차지하는 비율을 구한뒤 log를 취한 결과값입니다.

그리고 NLLLoss 는 negative likelihood loss의 약자로, 분류 모델에서 자주 쓰이는 loss function이고, 각 target에 해당하는 class들에 대한 수치가 있는 tensor(logsoftmax로 도출된 tensor)와, 각 target들에 해당하는 class index가 있는 tensor를 입력 받아서 loss를 계산합니다.

Adam optimizer는 momentum과 root mean square propagation의 두가지 기법이 적용된 optimizer로, momentum 개념을 활용해 gradient descent를 진행할때 momentum의 식을 따로 적용해서 weight update를 진행하게 되고, exponential moving average를 취해면서 weight update를 진행하게 됩니다.

3-2 LeNet-5 - optimizer, loss function function

```
#depending on the loss function, define criterion and choose whether softmax layer has to exist in the model
if loss_function_name=='CrossEntropyLoss':
    criterion=nn.CrossEntropyLoss().to(device)
    softmax_exist=False
elif loss_function_name=='MultiMarginLoss':
    criterion=nn.MultiMarginLoss().to(device)
    softmax_exist=False
elif loss_function_name=='MSELoss':
    criterion=nn.MSELoss().to(device)
    softmax_exist=True
```

```
#depending on the optimizer name, define optimizer
if optimizer_name=='Adam':
    optimizer=torch.optim.Adam(model.parameters(),lr=learning_rate)
elif optimizer_name=='SGD':
    optimizer=torch.optim.SGD(model.parameters(),lr=learning_rate)
elif optimizer_name=='RMSprop':
    optimizer=torch.optim.RMSprop(model.parameters(),lr=learning_rate)
```

다음은 LeNet-5 구조에서 사용한 loss function 과 optimizer 입니다. task2에서 training element를 analyze하기 위해서 다양한 loss function과 optimizer를 사용할 수 있도록 구현했습니다.

MultiMarginLoss에서는 각 class들에 해당하는 수치들이 있는 2-D tensor 와 각각의 target class 들이 있는 1-D tensor를 입력받아서, 서로간의 hinge loss를 구하는 방식으로 loss를 정의합니다. 여기서 hinge loss는 margin의 개념을 이용해서 구하는 loss 이며, 0 과 margin-target_index value-index_value 중 큰 값을 취하면서 계산하게 됩니다.

MSELoss input과 target 의 mean squared error를 loss로 정의하며, 코드에서는 모델에서 output data 로 나오는 각 이미지에 대한 크기 10짜리 벡터와, 이미지에 해당하는 index를 one-hot encoding으로 변환한 벡터값 사이의 MSELoss를 계산했습니다. MSELoss를 사용할 경우,target class의 one hot encoding tensor 와 model의 output tensor의 loss를 구하게 되는데, one hot encoding tensor 의 결과값이 0또는 1이므로, model의 output data 의 범위도 0에서 1사이의 값을 도출하도록 model의 마지막 layer에 softmax layer를 추가했습니다.

SGD optimizer은 data 를 1개를 본후 loss를 구해가면서 순차적으로 weight들을 업데이트 해주면서 loss function의 최솟값을 찾아가는 optimizer를 의미합니다.

RMSprop optimizer은 learning rate를 그대로 쓰지 않고, adaptive learning rate로 특정 수식을 통해 learning rate를 조정해 가면서 weight update를 진행하는 optimizer입니다.

```

for imgs, labels in train_data_loader:
    imgs=imgs.to(device)

    #depending on the loss function, change which the label format to use(index/one hot encoding)
    if loss_function_name=='CrossEntropyLoss':
        labels=labels.to(device)
    elif loss_function_name=='MultiMarginLoss':
        labels=labels.to(device)
    elif loss_function_name=='MSELoss':
        labels=torch.zeros(labels.shape[0],10).scatter_(1,labels.unsqueeze(1),1.).to(device)

    predictions=model(imgs)
    loss_function=criterion(predictions,labels)
    optimizer.zero_grad()
    loss_function.backward()
    optimizer.step()

```

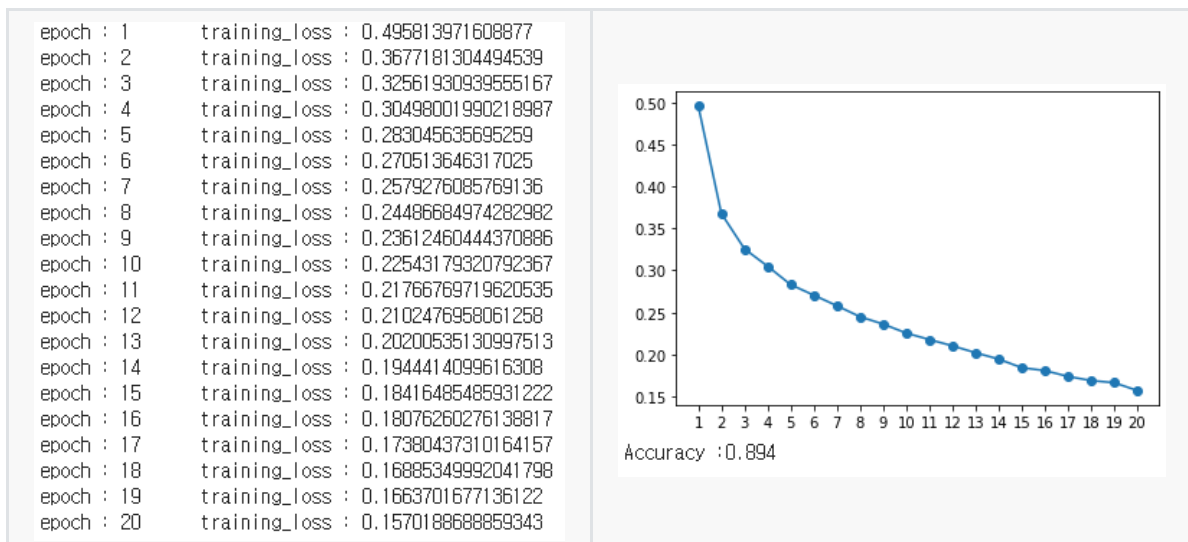
MSELoss를 사용할 경우, image 에 해당 하는 class index들을 one_hot encoding으로 변환한뒤 MSELoss를 계산했습니다.

3-3 optimizer and loss function concept

loss function 같은 경우, model에서 나온 output data와 target data의 차이의 지표를 나타내주며, loss function의 정의에 따라 data 에서 유사도를 나타내주는 지표를 바꿀 수있습니다. 이러한 loss function 을 정의해 output data와 target data의 다른 정도를 나타냈다면, optimizer는 model의 parameter들을 조정해 최종적으로 loss function의 값을 줄여주는 역할을 합니다. optimizer의 정의 에 따라, loss function의 최솟값을 찾아갈때 그 알고리즘과 경로가 달라지게 되고, 최소 loss function값에 도달하기 까지의 step 수가 달라지게 됩니다.

4. Screen-shot and analysis

4-1 Fully Connected Layer



다음은 Fully Connected Layer에서의 training loss, training loss graph와 test accuracy 입니다. Fully connected Layer를 학습 시킬때

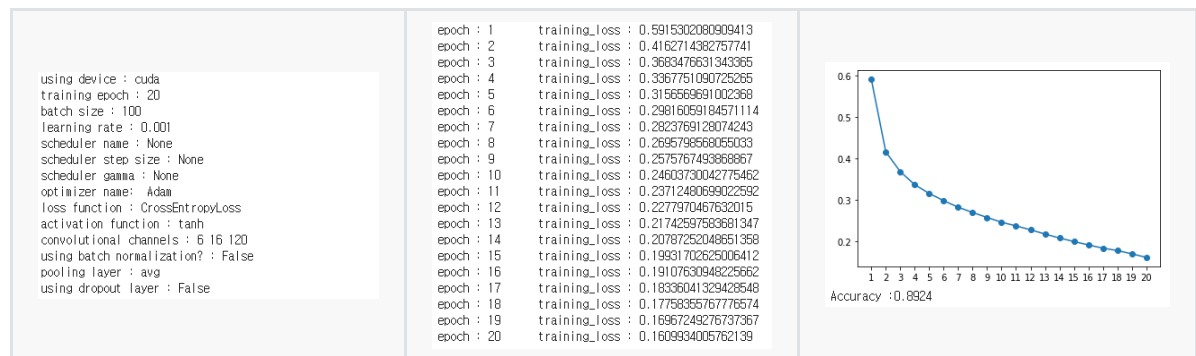
- training epoch : 20
- batch size : 100
- learning rate : 0.001

로 hyper parameter를 설정한 뒤 학습을 진행 했습니다. training loss 는 그래프에서 관찰한 바로는 계속 떨어지는 결과를 보였고, test data에 대한 accuracy는 89.4%로 나왔습니다.

4-2 LeNet-5

LeNet-5 구조에서는 base case 에서 model에 수정을 가하거나, hyper parameter에 변화를 주면서 학습을 진행해 봤고, 다음은 base case의 설정과 base case로 학습을 진행한 결과입니다.

- training epoch : 20
- batch size : 100
- learning rate : 0.001
- optimizer type : Adam
- loss function : CrossEntropyLoss
- activation function : tanh
- first, second, third channel in convolutional layers : 6,16,120
- pooling layer : average pooling layer
- no scheduler, no batch normalization, no dropout layer



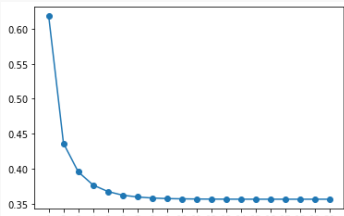
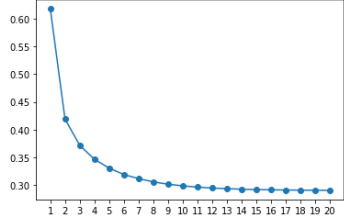
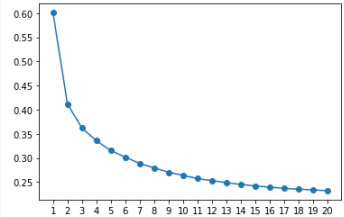
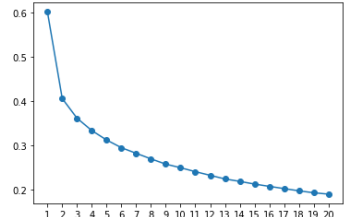
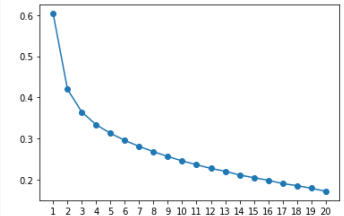
training loss 는 0.16 까지 떨어졌고, test data에 대한 accuracy는 89.24%로 나왔습니다.

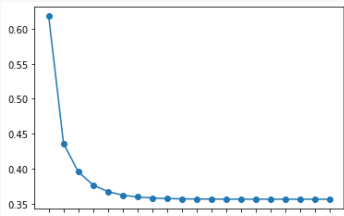
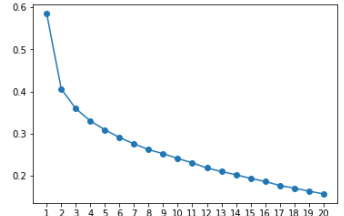
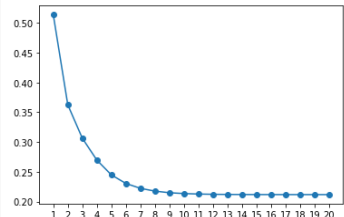
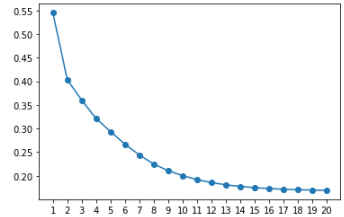
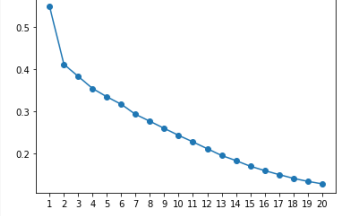
4-2-1 LeNet-5 optimizer parameter

optimizer parameter - scheduler

```
scheduler_name=None,
scheduler_step_size=None,
scheduler_gamma=None,
```

scheduler를 사용할경우, StepLR scheduler만 적용시켜 보았고, scheduler_step_size 의 data 마다 scheduler_gamma 값을 learning rate에 곱해서 적용시키도록 했습니다.

<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 100 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6180613595743976 epoch : 2 training_loss : 0.43632056931654606 epoch : 3 training_loss : 0.39551768327198 epoch : 4 training_loss : 0.37678616382181657 epoch : 5 training_loss : 0.367493970617655215 epoch : 6 training_loss : 0.3622086900720991 epoch : 7 training_loss : 0.3597026750942071 epoch : 8 training_loss : 0.35820335018138116 epoch : 9 training_loss : 0.3573904731372995 epoch : 10 training_loss : 0.3570248475422462 epoch : 11 training_loss : 0.35677012691895205 epoch : 12 training_loss : 0.3566447369505964 epoch : 13 training_loss : 0.356573834107713 epoch : 14 training_loss : 0.35653590070704605 epoch : 15 training_loss : 0.35651573670407116 epoch : 16 training_loss : 0.35650564812123786 epoch : 17 training_loss : 0.3564988116344212 epoch : 18 training_loss : 0.3564967001974581 epoch : 19 training_loss : 0.3564953451603647 epoch : 20 training_loss : 0.35649469271302253 </pre>	 <p>Accuracy :0.8607</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 200 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.617796214818955 epoch : 2 training_loss : 0.4191420417030655 epoch : 3 training_loss : 0.3716379365324978 epoch : 4 training_loss : 0.3463144936164223 epoch : 5 training_loss : 0.33052522264421 epoch : 6 training_loss : 0.31892224363982674 epoch : 7 training_loss : 0.3115380549679201 epoch : 8 training_loss : 0.3057482208063206 epoch : 9 training_loss : 0.3014621153722209 epoch : 10 training_loss : 0.2984754421561953 epoch : 11 training_loss : 0.2962576565782813 epoch : 12 training_loss : 0.29453512765467166 epoch : 13 training_loss : 0.29343517649918804 epoch : 14 training_loss : 0.2924738258868455 epoch : 15 training_loss : 0.29181324866891076 epoch : 16 training_loss : 0.2913357623666524 epoch : 17 training_loss : 0.29100527880092475 epoch : 18 training_loss : 0.290774941618244 epoch : 19 training_loss : 0.29059593732582176 epoch : 20 training_loss : 0.2904226682449327 </pre>	 <p>Accuracy :0.8795</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 400 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6009074200441435 epoch : 2 training_loss : 0.4115277966922849 epoch : 3 training_loss : 0.362090775561012 epoch : 4 training_loss : 0.33570592286686113 epoch : 5 training_loss : 0.31518347191313933 epoch : 6 training_loss : 0.30184568387766675 epoch : 7 training_loss : 0.28870746694505167 epoch : 8 training_loss : 0.279492983594537 epoch : 9 training_loss : 0.2703416785846151 epoch : 10 training_loss : 0.26413932943095775 epoch : 11 training_loss : 0.25736740607788294 epoch : 12 training_loss : 0.25297449606160316 epoch : 13 training_loss : 0.24898778229554473 epoch : 14 training_loss : 0.24517679636677112 epoch : 15 training_loss : 0.24218726866506832 epoch : 16 training_loss : 0.2396431883300346 epoch : 17 training_loss : 0.23707691097011188 epoch : 18 training_loss : 0.23531543149302386 epoch : 19 training_loss : 0.23380202237518656 epoch : 20 training_loss : 0.23226914074271915 </pre>	 <p>Accuracy :0.8926</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 800 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6019727107385795 epoch : 2 training_loss : 0.4061948098242281 epoch : 3 training_loss : 0.3617515151699383 epoch : 4 training_loss : 0.33379805967870666 epoch : 5 training_loss : 0.3125373695359958 epoch : 6 training_loss : 0.29510984801918655 epoch : 7 training_loss : 0.28240511449674766 epoch : 8 training_loss : 0.2698678268864747 epoch : 9 training_loss : 0.25799579880340876 epoch : 10 training_loss : 0.24885956771920131 epoch : 11 training_loss : 0.2409665040663917 epoch : 12 training_loss : 0.232711381527285 epoch : 13 training_loss : 0.22453995811062096 epoch : 14 training_loss : 0.21903233555455998 epoch : 15 training_loss : 0.212949556224048 epoch : 16 training_loss : 0.20775208875530952 epoch : 17 training_loss : 0.2027231671288609 epoch : 18 training_loss : 0.19772416987766814 epoch : 19 training_loss : 0.19329462370524808 epoch : 20 training_loss : 0.19022944287707405 </pre>	 <p>Accuracy :0.8946</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 1600 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6033764415979387 epoch : 2 training_loss : 0.41978302587406566 epoch : 3 training_loss : 0.36431549847126016 epoch : 4 training_loss : 0.33376376971602434 epoch : 5 training_loss : 0.3127956360414955 epoch : 6 training_loss : 0.2955114412347477 epoch : 7 training_loss : 0.2810743123417097 epoch : 8 training_loss : 0.26826874271035206 epoch : 9 training_loss : 0.25647145781668053 epoch : 10 training_loss : 0.24589060942331936 epoch : 11 training_loss : 0.23660376476744807 epoch : 12 training_loss : 0.22760236504177248 epoch : 13 training_loss : 0.22081578056017537 epoch : 14 training_loss : 0.21168880626717434 epoch : 15 training_loss : 0.20507679077486204 epoch : 16 training_loss : 0.19888210850457336 epoch : 17 training_loss : 0.19080937479482093 epoch : 18 training_loss : 0.18584976746390267 epoch : 19 training_loss : 0.17958423726260653 epoch : 20 training_loss : 0.17201191214844602 </pre>	 <p>Accuracy :0.8955</p>

<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 100 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization?: False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.6180613595743976 epoch : 2 training_loss : 0.43632056931654606 epoch : 3 training_loss : 0.39551768327198 epoch : 4 training_loss : 0.37678516382181657 epoch : 5 training_loss : 0.367493970617655215 epoch : 6 training_loss : 0.3622086900720991 epoch : 7 training_loss : 0.3597026750942071 epoch : 8 training_loss : 0.35820335018138116 epoch : 9 training_loss : 0.3573904731372995 epoch : 10 training_loss : 0.3570248475422462 epoch : 11 training_loss : 0.35677012691895205 epoch : 12 training_loss : 0.3566447369505964 epoch : 13 training_loss : 0.3565733834107713 epoch : 14 training_loss : 0.35653590070704605 epoch : 15 training_loss : 0.35651573670407116 epoch : 16 training_loss : 0.35650564812123786 epoch : 17 training_loss : 0.3564988116344212 epoch : 18 training_loss : 0.3564967001974581 epoch : 19 training_loss : 0.3564953451603647 epoch : 20 training_loss : 0.35649469271302253 </pre>	 <p>Accuracy : 0.8607</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 3200 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization?: False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.5840853226681562 epoch : 2 training_loss : 0.40582744434475904 epoch : 3 training_loss : 0.35916734586159355 epoch : 4 training_loss : 0.33010780458648975 epoch : 5 training_loss : 0.30905974050362917 epoch : 6 training_loss : 0.29074672939327256 epoch : 7 training_loss : 0.2757790904492141 epoch : 8 training_loss : 0.26213314200441057 epoch : 9 training_loss : 0.2524222317834697 epoch : 10 training_loss : 0.24125636408726361 epoch : 11 training_loss : 0.23075525065263092 epoch : 12 training_loss : 0.21877840841809917 epoch : 13 training_loss : 0.21014988731513831 epoch : 14 training_loss : 0.2026900933496654 epoch : 15 training_loss : 0.19393617336327815 epoch : 16 training_loss : 0.186825845633856 epoch : 17 training_loss : 0.1771175915934148 epoch : 18 training_loss : 0.17095057719076667 epoch : 19 training_loss : 0.1639133651244143 epoch : 20 training_loss : 0.15728809435230987 </pre>	 <p>Accuracy : 0.8943</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 100 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization?: False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.513833322107791 epoch : 2 training_loss : 0.3625050102174282 epoch : 3 training_loss : 0.30642495313038415 epoch : 4 training_loss : 0.26976980914672233 epoch : 5 training_loss : 0.24482273253301756 epoch : 6 training_loss : 0.23049245174974192 epoch : 7 training_loss : 0.22226559723416975 epoch : 8 training_loss : 0.21761161107569935 epoch : 9 training_loss : 0.2148761066173514 epoch : 10 training_loss : 0.21349653581778202 epoch : 11 training_loss : 0.21270909768839652 epoch : 12 training_loss : 0.21225793966402612 epoch : 13 training_loss : 0.21203059827287996 epoch : 14 training_loss : 0.2119038633605505 epoch : 15 training_loss : 0.21183087677757 epoch : 16 training_loss : 0.211796317907671 epoch : 17 training_loss : 0.21177681372811372 epoch : 18 training_loss : 0.211764142449695 epoch : 19 training_loss : 0.21176063152550973 epoch : 20 training_loss : 0.21175768737991654 </pre>	 <p>Accuracy : 0.8906</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 200 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization?: False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.5453331395238639 epoch : 2 training_loss : 0.40354852688809234 epoch : 3 training_loss : 0.35991684012115 epoch : 4 training_loss : 0.3210926263406872 epoch : 5 training_loss : 0.2935461398462459 epoch : 6 training_loss : 0.2667821898909095 epoch : 7 training_loss : 0.24409307700892309 epoch : 8 training_loss : 0.22471638689196757 epoch : 9 training_loss : 0.21071226153522746 epoch : 10 training_loss : 0.20020563216259105 epoch : 11 training_loss : 0.1918032628124895 epoch : 12 training_loss : 0.18550014302134527 epoch : 13 training_loss : 0.1808000300949819 epoch : 14 training_loss : 0.1772666180744762 epoch : 15 training_loss : 0.17470217343419786 epoch : 16 training_loss : 0.1725355091739735 epoch : 17 training_loss : 0.17141085634628553 epoch : 18 training_loss : 0.17039464818313704 epoch : 19 training_loss : 0.16962294563651073 epoch : 20 training_loss : 0.16909348543733357 </pre>	 <p>Accuracy : 0.8881</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 400 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization?: False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.5480396189788981 epoch : 2 training_loss : 0.4112331627806025 epoch : 3 training_loss : 0.36226723673443036 epoch : 4 training_loss : 0.35377364059289335 epoch : 5 training_loss : 0.339985593893925 epoch : 6 training_loss : 0.3162764209508894 epoch : 7 training_loss : 0.2923758566503723 epoch : 8 training_loss : 0.27601169555137595 epoch : 9 training_loss : 0.259106130823493 epoch : 10 training_loss : 0.24268633543203297 epoch : 11 training_loss : 0.22710791987677412 epoch : 12 training_loss : 0.21100976332401228 epoch : 13 training_loss : 0.1949148685485125 epoch : 14 training_loss : 0.1828224262036385 epoch : 15 training_loss : 0.16938873846083874 epoch : 16 training_loss : 0.1591241216969987 epoch : 17 training_loss : 0.14967562403629885 epoch : 18 training_loss : 0.1407130939451358 epoch : 19 training_loss : 0.1336487934365869 epoch : 20 training_loss : 0.1277080815533797 </pre>	 <p>Accuracy : 0.8852</p>

<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : StepLR scheduler step size : 100 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6180613595743976 epoch : 2 training_loss : 0.43632056931654606 epoch : 3 training_loss : 0.39551768327198 epoch : 4 training_loss : 0.37678616382181657 epoch : 5 training_loss : 0.36749397061765215 epoch : 6 training_loss : 0.3622086900720991 epoch : 7 training_loss : 0.3597026750942071 epoch : 8 training_loss : 0.35820335018138116 epoch : 9 training_loss : 0.3573904731372995 epoch : 10 training_loss : 0.3570248475422462 epoch : 11 training_loss : 0.35677012691895205 epoch : 12 training_loss : 0.3566447369505964 epoch : 13 training_loss : 0.3565733834107713 epoch : 14 training_loss : 0.35653590070704605 epoch : 15 training_loss : 0.35651573670407116 epoch : 16 training_loss : 0.35650564812123786 epoch : 17 training_loss : 0.3564988116344212 epoch : 18 training_loss : 0.3564967001974581 epoch : 19 training_loss : 0.3564953451603647 epoch : 20 training_loss : 0.35649469271302253 </pre>	<p>Accuracy : 0.8607</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 800 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.5762661560375486 epoch : 2 training_loss : 0.437494908620914 epoch : 3 training_loss : 0.41599478011329993 epoch : 4 training_loss : 0.39315080448965057 epoch : 5 training_loss : 0.3786434741318223 epoch : 6 training_loss : 0.3630796959495176 epoch : 7 training_loss : 0.3461530820777022 epoch : 8 training_loss : 0.34075666675964994 epoch : 9 training_loss : 0.3234682255486645 epoch : 10 training_loss : 0.30981657023231135 epoch : 11 training_loss : 0.30257535770535443 epoch : 12 training_loss : 0.287911506653153 epoch : 13 training_loss : 0.2762016362076008 epoch : 14 training_loss : 0.2684878108153737 epoch : 15 training_loss : 0.2588835526208083 epoch : 16 training_loss : 0.2468542345909324 epoch : 17 training_loss : 0.2360386505324112 epoch : 18 training_loss : 0.22604241576523266 epoch : 19 training_loss : 0.21936515361587543 epoch : 20 training_loss : 0.20488230145225922 </pre>	<p>Accuracy : 0.8862</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 1600 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.5362535458068113 epoch : 2 training_loss : 0.43994408744076824 epoch : 3 training_loss : 0.42678880838056443 epoch : 4 training_loss : 0.4107832701255881 epoch : 5 training_loss : 0.410339961854577 epoch : 6 training_loss : 0.3982089239856085 epoch : 7 training_loss : 0.394315097903212 epoch : 8 training_loss : 0.37685591772198673 epoch : 9 training_loss : 0.3729591053972643 epoch : 10 training_loss : 0.36704116749266763 epoch : 11 training_loss : 0.3705488968392211 epoch : 12 training_loss : 0.35410534881065394 epoch : 13 training_loss : 0.3569569425781568 epoch : 14 training_loss : 0.347900962459341 epoch : 15 training_loss : 0.3377361296241481 epoch : 16 training_loss : 0.32871527810891515 epoch : 17 training_loss : 0.32740839851399256 epoch : 18 training_loss : 0.3190077110379364 epoch : 19 training_loss : 0.3220106979086994 epoch : 20 training_loss : 0.30791878581047066 </pre>	<p>Accuracy : 0.8711</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.01 scheduler name : StepLR scheduler step size : 3200 scheduler gamma : 0.9 optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.5789371017615002 epoch : 2 training_loss : 0.435064281870921 epoch : 3 training_loss : 0.4264937089631953 epoch : 4 training_loss : 0.4162152802447481 epoch : 5 training_loss : 0.4147975791742406 epoch : 6 training_loss : 0.4150200142959753 epoch : 7 training_loss : 0.39544710499544933 epoch : 8 training_loss : 0.3934672372788193 epoch : 9 training_loss : 0.398398539324603 epoch : 10 training_loss : 0.39226419255137474 epoch : 11 training_loss : 0.392473282366991 epoch : 12 training_loss : 0.38582619709273197 epoch : 13 training_loss : 0.376007248386741 epoch : 14 training_loss : 0.3719922996312382 epoch : 15 training_loss : 0.37676554709339135 epoch : 16 training_loss : 0.37712338695823486 epoch : 17 training_loss : 0.36231858444710563 epoch : 18 training_loss : 0.3564924982438486 epoch : 19 training_loss : 0.35346671638905896 epoch : 20 training_loss : 0.35882335826754647 </pre>	<p>Accuracy : 0.8522</p>

다음은 scheduler를 사용해서 learning rate를 조정해본 결과입니다. 결과를 순차적으로 나열해보면

scheduler name : StepLR

scheduler gamma : 0.9

initial learning rate : 0.001일때

- scheduler_step_size=100 -> 최종 training loss : 0.3564, accuracy : 86.07%
- scheduler_step_size=200 -> 최종 training loss : 0.2904, accuracy : 87.95%
- scheduler_step_size=400 -> 최종 training loss : 0.2322, accuracy : 89.26%
- scheduler_step_size=800 -> 최종 training loss : 0.1902, accuracy : 89.46%
- scheduler_step_size=1600 -> 최종 training loss : 0.1720, accuracy : 89.55%
- scheduler_step_size=3200 -> 최종 training loss : 0.157264, accuracy : 89.43%

scheduler name : StepLR

scheduler gamma : 0.9

initial learning rate : 0.01 일때

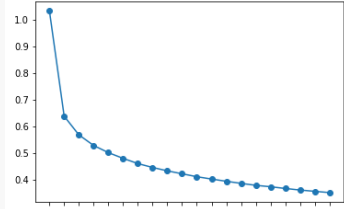
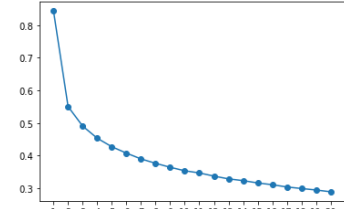
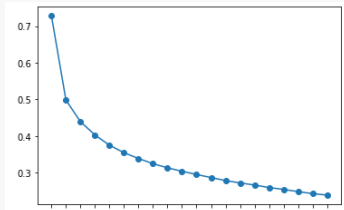
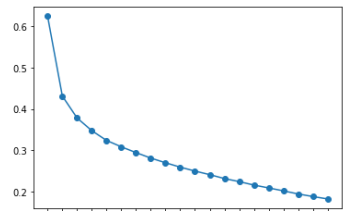
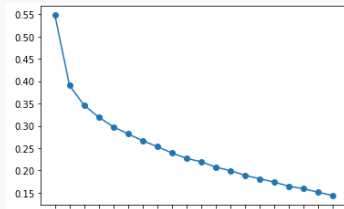
- scheduler_step_size=100 -> 최종 training loss : 0.2117, accuracy : 89.06%
- scheduler_step_size=200 -> 최종 training loss : 0.1690, accuracy : 88.81%
- scheduler_step_size=400 -> 최종 training loss : 0.1277, accuracy : 88.52%
- scheduler_step_size=800 -> 최종 training loss : 0.2048, accuracy : 88.62%
- scheduler_step_size=1600 -> 최종 training loss : 0.3079, accuracy : 87.11%
- scheduler_step_size=3200 -> 최종 training loss : 0.3588, accuracy : 85.22%

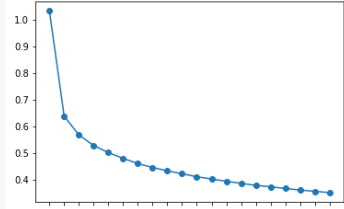
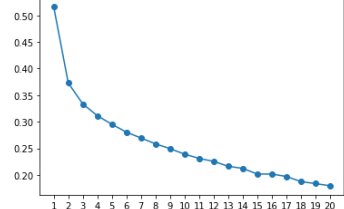
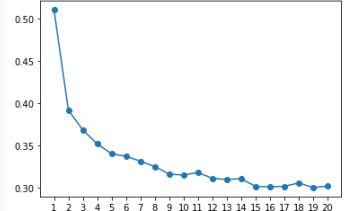
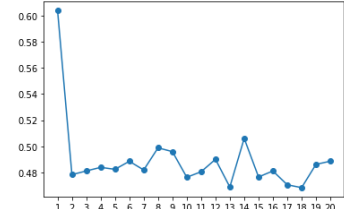
와 같이 최종 training loss 와 accuracy가 나왔습니다. 초기 learning rate가 0.001일때, scheduler step size를 100에서부터 시작해 3200까지 2배씩 증가시켜본 결과, 초기에 training loss 는 점차 줄어들었고, accuracy 는 점차 올라가다가 더이상 안올라가는 양상을 보였습니다. 초기 learning rate가 0.01 일때는, 동일하게 scheduler step size를 증가시켜본 결과, loss 는 내려갔다가 올라갔으며, accuracy는 점차 내려가는 양상을 보였습니다.

optimizer parameter - learning rate

```
learning_rate=0.001,
```

기본적으로 learning rate는 0.001 로 설정되었으며, learning rate를 조정해 가며 학습양상을 관찰해 보았습니다.

<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 1.032063073714574 epoch : 2 training_loss : 0.6369053021570044 epoch : 3 training_loss : 0.5668357674777502 epoch : 4 training_loss : 0.527209066649278 epoch : 5 training_loss : 0.5002740390102068 epoch : 6 training_loss : 0.4787459927797322 epoch : 7 training_loss : 0.4592682601227116 epoch : 8 training_loss : 0.44468780030806815 epoch : 9 training_loss : 0.4319568471858895 epoch : 10 training_loss : 0.4209667780001963 epoch : 11 training_loss : 0.4100122858087219 epoch : 12 training_loss : 0.401229893043637 epoch : 13 training_loss : 0.39285698021451626 epoch : 14 training_loss : 0.38502286953230674 epoch : 15 training_loss : 0.3776866511752208 epoch : 16 training_loss : 0.3723484473675492 epoch : 17 training_loss : 0.3659622803330421 epoch : 18 training_loss : 0.36001117100318275 epoch : 19 training_loss : 0.35517257582396267 epoch : 20 training_loss : 0.35039760063091957 </pre>	 <p>Accuracy : 0.9624</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0002 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.8437147465844952 epoch : 2 training_loss : 0.5503435026109216 epoch : 3 training_loss : 0.490643637602528 epoch : 4 training_loss : 0.453689357837042 epoch : 5 training_loss : 0.42729324343303826 epoch : 6 training_loss : 0.4077813456952572 epoch : 7 training_loss : 0.3899595027323577 epoch : 8 training_loss : 0.3765505272895096 epoch : 9 training_loss : 0.36433258824050424 epoch : 10 training_loss : 0.3536464362343153 epoch : 11 training_loss : 0.34702548551062806 epoch : 12 training_loss : 0.33730529072384025 epoch : 13 training_loss : 0.3287838586171467 epoch : 14 training_loss : 0.323426641970873 epoch : 15 training_loss : 0.3163192129631835 epoch : 16 training_loss : 0.310990120763983 epoch : 17 training_loss : 0.30397971528271944 epoch : 18 training_loss : 0.2992559308070584 epoch : 19 training_loss : 0.2946536358073352 epoch : 20 training_loss : 0.28921771885206277 </pre>	 <p>Accuracy : 0.8757</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0004 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.7279877336323257 epoch : 2 training_loss : 0.4982179389403897 epoch : 3 training_loss : 0.43954919504622614 epoch : 4 training_loss : 0.40307281548778245 epoch : 5 training_loss : 0.3754351506382226 epoch : 6 training_loss : 0.3548885261764129 epoch : 7 training_loss : 0.3391618776321415 epoch : 8 training_loss : 0.32490739926695805 epoch : 9 training_loss : 0.31368492633104345 epoch : 10 training_loss : 0.30428190703193353 epoch : 11 training_loss : 0.29558991183837235 epoch : 12 training_loss : 0.28713425949215915 epoch : 13 training_loss : 0.2789660481363537 epoch : 14 training_loss : 0.27251460189620635 epoch : 15 training_loss : 0.2667561991140247 epoch : 16 training_loss : 0.25995500185837345 epoch : 17 training_loss : 0.2548031396915512 epoch : 18 training_loss : 0.24879641775041822 epoch : 19 training_loss : 0.24336173319568247 epoch : 20 training_loss : 0.23945707768201824 </pre>	 <p>Accuracy : 0.8903</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0008 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.6240380738675595 epoch : 2 training_loss : 0.4312675282110773 epoch : 3 training_loss : 0.3782778915514545 epoch : 4 training_loss : 0.3476080599923932 epoch : 5 training_loss : 0.3240703739970925 epoch : 6 training_loss : 0.3086542385444044 epoch : 7 training_loss : 0.2946450110773244 epoch : 8 training_loss : 0.2814803546170393 epoch : 9 training_loss : 0.2701596002156536 epoch : 10 training_loss : 0.2595077771569295 epoch : 11 training_loss : 0.25011661530782797 epoch : 12 training_loss : 0.24144058949003624 epoch : 13 training_loss : 0.23197399746626624 epoch : 14 training_loss : 0.2246197825918596 epoch : 15 training_loss : 0.21630376479277996 epoch : 16 training_loss : 0.2093539890088142 epoch : 17 training_loss : 0.20229337708626203 epoch : 18 training_loss : 0.19484665869424764 epoch : 19 training_loss : 0.1869833812961966 epoch : 20 training_loss : 0.18282919514924267 </pre>	 <p>Accuracy : 0.8939</p>
<p>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0016 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.5478910033202174 epoch : 2 training_loss : 0.3906633410811422 epoch : 3 training_loss : 0.34572349262734264 epoch : 4 training_loss : 0.31956391602754614 epoch : 5 training_loss : 0.29752632066607465 epoch : 6 training_loss : 0.28208616218219185 epoch : 7 training_loss : 0.26676336842278664 epoch : 8 training_loss : 0.25337862440695363 epoch : 9 training_loss : 0.23946059420704852 epoch : 10 training_loss : 0.2271236135832864 epoch : 11 training_loss : 0.2195812836786112 epoch : 12 training_loss : 0.20770837976286802 epoch : 13 training_loss : 0.1996976747736333 epoch : 14 training_loss : 0.18928507857024682 epoch : 15 training_loss : 0.18150571058193857 epoch : 16 training_loss : 0.1740712713760631 epoch : 17 training_loss : 0.16463001876448569 epoch : 18 training_loss : 0.15928796063487718 epoch : 19 training_loss : 0.15138123572493573 epoch : 20 training_loss : 0.14390666071946426 </pre>	 <p>Accuracy : 0.8931</p>

<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 1.032063073714574 epoch : 2 training_loss : 0.6369053021570044 epoch : 3 training_loss : 0.5668357674777502 epoch : 4 training_loss : 0.527209066649278 epoch : 5 training_loss : 0.5002740390102068 epoch : 6 training_loss : 0.4787459927797322 epoch : 7 training_loss : 0.4592682601227116 epoch : 8 training_loss : 0.44468780030806815 epoch : 9 training_loss : 0.4319568471858895 epoch : 10 training_loss : 0.4209667780001963 epoch : 11 training_loss : 0.4100122858087219 epoch : 12 training_loss : 0.401229893043637 epoch : 13 training_loss : 0.39285698021451626 epoch : 14 training_loss : 0.38502286953230674 epoch : 15 training_loss : 0.3776866511752208 epoch : 16 training_loss : 0.3723484473675492 epoch : 17 training_loss : 0.365962280330421 epoch : 18 training_loss : 0.36001117100318275 epoch : 19 training_loss : 0.35517257582396267 epoch : 20 training_loss : 0.3503976063091957 </pre>	 <p>Accuracy : 0.9624</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0032 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.51618105083704 epoch : 2 training_loss : 0.3727728532503047 epoch : 3 training_loss : 0.33347545241316157 epoch : 4 training_loss : 0.31119157078365495 epoch : 5 training_loss : 0.29523323970536414 epoch : 6 training_loss : 0.28027311979482594 epoch : 7 training_loss : 0.2694248032694065 epoch : 8 training_loss : 0.2583569201049211 epoch : 9 training_loss : 0.24935807020713993 epoch : 10 training_loss : 0.23889163520187143 epoch : 11 training_loss : 0.23097965019444633 epoch : 12 training_loss : 0.2252948959938628 epoch : 13 training_loss : 0.21622754838317618 epoch : 14 training_loss : 0.21213956707467635 epoch : 15 training_loss : 0.2015566669288863 epoch : 16 training_loss : 0.20158720046281814 epoch : 17 training_loss : 0.1969477967607477 epoch : 18 training_loss : 0.18733991768300984 epoch : 19 training_loss : 0.18360985551650322 epoch : 20 training_loss : 0.17946156968052215 </pre>	 <p>Accuracy : 0.8874</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0064 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.5100683987271 epoch : 2 training_loss : 0.3915613585462171 epoch : 3 training_loss : 0.3682814854631821 epoch : 4 training_loss : 0.35174130196372677 epoch : 5 training_loss : 0.3401208450893565 epoch : 6 training_loss : 0.33717375857134657 epoch : 7 training_loss : 0.331379687358699 epoch : 8 training_loss : 0.3250195976595086 epoch : 9 training_loss : 0.3160669191430014 epoch : 10 training_loss : 0.3150151549528047 epoch : 11 training_loss : 0.3179951080059009 epoch : 12 training_loss : 0.3114630840718727 epoch : 13 training_loss : 0.3096816489348811 epoch : 14 training_loss : 0.31080475289132947 epoch : 15 training_loss : 0.3015133107701935 epoch : 16 training_loss : 0.30113017693161925 epoch : 17 training_loss : 0.301620692665928 epoch : 18 training_loss : 0.3057328577091297 epoch : 19 training_loss : 0.3001993296792109 epoch : 20 training_loss : 0.3018894484266636 </pre>	 <p>Accuracy : 0.8741</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.0128 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6037508361041537 epoch : 2 training_loss : 0.4783688039580979 epoch : 3 training_loss : 0.4812037942806878 epoch : 4 training_loss : 0.4839437962075072 epoch : 5 training_loss : 0.4824469013015427 epoch : 6 training_loss : 0.48858636289834956 epoch : 7 training_loss : 0.4819534145792321 epoch : 8 training_loss : 0.4988680599133176 epoch : 9 training_loss : 0.49595303369065086 epoch : 10 training_loss : 0.47638089433312397 epoch : 11 training_loss : 0.4805862305810052 epoch : 12 training_loss : 0.49009065931042073 epoch : 13 training_loss : 0.4689386853079008 epoch : 14 training_loss : 0.5060998659626641 epoch : 15 training_loss : 0.4766086196154362 epoch : 16 training_loss : 0.48117769663532545 epoch : 17 training_loss : 0.47063931167125717 epoch : 18 training_loss : 0.46843952129284544 epoch : 19 training_loss : 0.4862117742747075 epoch : 20 training_loss : 0.48866436099012733 </pre>	 <p>Accuracy : 0.8136</p>

다음은 learning rate를 조정해가며 training loss 와 accuracy를 관찰해본 결과입니다. 결과를 순차적으로 나열해보면,

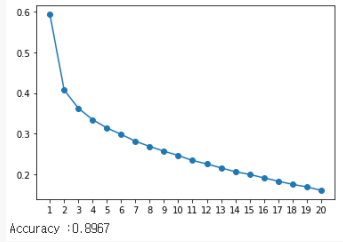
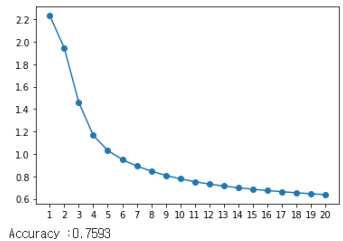
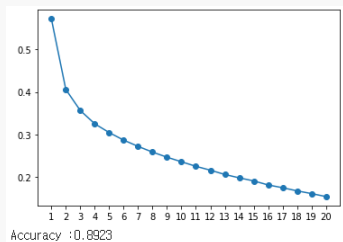
- learning rate : 0.0001 -> 최종 training loss : 0.3503, accuracy : 86.24%
- learning rate : 0.0002 -> 최종 training loss : 0.2892, accuracy : 87.57%
- learning rate : 0.0004 -> 최종 training loss : 0.2394, accuracy : 89.03%
- learning rate : 0.0008 -> 최종 training loss : 0.1828, accuracy : 89.39%
- learning rate : 0.0016 -> 최종 training loss : 0.1439, accuracy : 89.31%
- learning rate : 0.0032 -> 최종 training loss : 0.1794, accuracy : 88.74%
- learning rate : 0.0064 -> 최종 training loss : 0.3018, accuracy : 87.41%
- learning rate : 0.0128 -> 최종 training loss : 0.4886, accuracy : 81.36%

learning rate를 0.0001서부터 0.0128까지 2배씩 증가시키면서 실행을 시켰으며, training loss 는 내려가다가 올라가는 양상을, 그리고 accuracy 는 올라가다가 내려가는 양상을 보였습니다.

optimizer parameter - optimizer type

```
optimizer_name='Adam',
```

기본적으로 Adam optimizer가 설정되어 있고, SGD, RMSprop을 사용할 수 있습니다.

<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 0.5937748376031718 epoch : 2 training_loss : 0.4075403103480738 epoch : 3 training_loss : 0.3624006781478723 epoch : 4 training_loss : 0.33420978911220965 epoch : 5 training_loss : 0.31375193335115936 epoch : 6 training_loss : 0.29790582684179134 epoch : 7 training_loss : 0.2810117998470863 epoch : 8 training_loss : 0.26820459458977 epoch : 9 training_loss : 0.2564443933342893 epoch : 10 training_loss : 0.24587894907842087 epoch : 11 training_loss : 0.23359458415458606 epoch : 12 training_loss : 0.22519160710275185 epoch : 13 training_loss : 0.2153947535778085 epoch : 14 training_loss : 0.2058167903746168 epoch : 15 training_loss : 0.1993262519190707 epoch : 16 training_loss : 0.19068713477502283 epoch : 17 training_loss : 0.18262564335639278 epoch : 18 training_loss : 0.17488193122049162 epoch : 19 training_loss : 0.16840954909722022 epoch : 20 training_loss : 0.1601835867762567</pre>	 <p>Accuracy : 0.8967</p>
<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: SGD loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 2.2327298816045116 epoch : 2 training_loss : 1.9449730346600214 epoch : 3 training_loss : 1.4606297514836 epoch : 4 training_loss : 1.1686673982739438 epoch : 5 training_loss : 1.0310135473807647 epoch : 6 training_loss : 0.950175972680251 epoch : 7 training_loss : 0.892120480394778 epoch : 8 training_loss : 0.846483267347018 epoch : 9 training_loss : 0.80916843568319648 epoch : 10 training_loss : 0.7782651387324326 epoch : 11 training_loss : 0.7526199369701705 epoch : 12 training_loss : 0.731503544251124 epoch : 13 training_loss : 0.713746660699446 epoch : 14 training_loss : 0.6985744018356003 epoch : 15 training_loss : 0.6854497174421947 epoch : 16 training_loss : 0.673978398272743 epoch : 17 training_loss : 0.6633774288992091 epoch : 18 training_loss : 0.654009890023278 epoch : 19 training_loss : 0.6453037180006502 epoch : 20 training_loss : 0.6372503655824465</pre>	 <p>Accuracy : 0.7593</p>
<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: RMSprop loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 0.5719735958675547 epoch : 2 training_loss : 0.40593696157137527 epoch : 3 training_loss : 0.3564660529295607 epoch : 4 training_loss : 0.325313443802297 epoch : 5 training_loss : 0.3045962254997095 epoch : 6 training_loss : 0.2869374294703205 epoch : 7 training_loss : 0.2721185425964246 epoch : 8 training_loss : 0.2597106309080835 epoch : 9 training_loss : 0.24672357937941994 epoch : 10 training_loss : 0.2360025750600584 epoch : 11 training_loss : 0.2253272504359481 epoch : 12 training_loss : 0.2161798460036519 epoch : 13 training_loss : 0.2061570717953146 epoch : 14 training_loss : 0.1991624417752025 epoch : 15 training_loss : 0.19099148685733475 epoch : 16 training_loss : 0.18150734265645355 epoch : 17 training_loss : 0.17498086813837269 epoch : 18 training_loss : 0.16761784157406302 epoch : 19 training_loss : 0.16122557455673822 epoch : 20 training_loss : 0.15419460770984497</pre>	 <p>Accuracy : 0.8923</p>

각각의 optimizer를 사용한 결과들을 나열해보면,

- Adam optimizer -> 최종 training loss : 0.1601, accuracy : 89.67%
- SGD optimizer -> 최종 training loss : 0.6372, accuracy : 75.93%
- RMSprop optimizer -> 최종 training loss : 0.1541, accuracy : 89.23%

로 결과가 나왔고, Adam optimizer 를 이용할때, 가장 높은 accuracy가 나왔고, SGD optimizer에서 가장 낮은 최종 training loss와 accuracy, 그리고 RMS prop에서는 가장 낮은 최종 training loss와 두번째로 낮은 accuracy가 나왔습니다.

4-2-2 LeNet-5 loss function

```
loss_function_name='CrossEntropyLoss',
```

기본적으로 loss function은 CrossEntropyLoss로 설정되어 있고, MSELoss, MultiMarginLoss를 사용할 수 있습니다.

<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.6065581858654817 epoch : 2 training_loss : 0.41584018650154314 epoch : 3 training_loss : 0.36662216340502063 epoch : 4 training_loss : 0.33611698463659175 epoch : 5 training_loss : 0.3132741793245079 epoch : 6 training_loss : 0.2982433762152989 epoch : 7 training_loss : 0.2817628322045007 epoch : 8 training_loss : 0.2701565208286047 epoch : 9 training_loss : 0.25979104931155816 epoch : 10 training_loss : 0.24856663712610821 epoch : 11 training_loss : 0.2383883121858041 epoch : 12 training_loss : 0.22844507317990068 epoch : 13 training_loss : 0.21938018186638752 epoch : 14 training_loss : 0.21122658912092454 epoch : 15 training_loss : 0.20111725792288765 epoch : 16 training_loss : 0.19570764139294616 epoch : 17 training_loss : 0.1887988485147554 epoch : 18 training_loss : 0.18004461145649336 epoch : 19 training_loss : 0.1732385204865693 epoch : 20 training_loss : 0.16828994470338018 </pre>	<p>Accuracy : 0.8971</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : MSELoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.030483113691831654 epoch : 2 training_loss : 0.021455154546226055 epoch : 3 training_loss : 0.01896900478595247 epoch : 4 training_loss : 0.01756045123329385 epoch : 5 training_loss : 0.016512467847205684 epoch : 6 training_loss : 0.015786007408631115 epoch : 7 training_loss : 0.015072220380728427 epoch : 8 training_loss : 0.014423504730220893 epoch : 9 training_loss : 0.01393474506912753 epoch : 10 training_loss : 0.01330185066830878 epoch : 11 training_loss : 0.012795333752874293 epoch : 12 training_loss : 0.012378946219493314 epoch : 13 training_loss : 0.01193595226182671 epoch : 14 training_loss : 0.011395215804999055 epoch : 15 training_loss : 0.0111103616367711506 epoch : 16 training_loss : 0.010553518096373098 epoch : 17 training_loss : 0.01031118895480858 epoch : 18 training_loss : 0.009826179776573558 epoch : 19 training_loss : 0.009543605901999402 epoch : 20 training_loss : 0.009166513123588326 </pre>	<p>Accuracy : 0.8898</p>
<pre> using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : MultiMarginLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.10083147623265772 epoch : 2 training_loss : 0.059637299346799674 epoch : 3 training_loss : 0.0509464248704692 epoch : 4 training_loss : 0.04586576329078521 epoch : 5 training_loss : 0.042018924987254035 epoch : 6 training_loss : 0.03880912058055404 epoch : 7 training_loss : 0.03657477573336408 epoch : 8 training_loss : 0.03424897536014517 epoch : 9 training_loss : 0.03244297993990277 epoch : 10 training_loss : 0.030675030816346437 epoch : 11 training_loss : 0.02914304786206536 epoch : 12 training_loss : 0.02776318799083434 epoch : 13 training_loss : 0.026605369338455285 epoch : 14 training_loss : 0.025172571859632942 epoch : 15 training_loss : 0.024208037341013525 epoch : 16 training_loss : 0.023100629476054266 epoch : 17 training_loss : 0.022230433992420615 epoch : 18 training_loss : 0.02125905371115853 epoch : 19 training_loss : 0.020615759890836957 epoch : 20 training_loss : 0.019494422130131497 </pre>	<p>Accuracy : 0.8962</p>

각각의 loss function들에 대한 결과들을 나열해보면

- CrossEntropyLoss -> 최종 training loss : 0.1682, accuracy : 89.71%
- MSELoss-> 최종 training loss : 0.0091, accuracy : 88.98%
- MultiMarginLoss-> 최종 training loss : 0.0194, accuracy : 89.62%

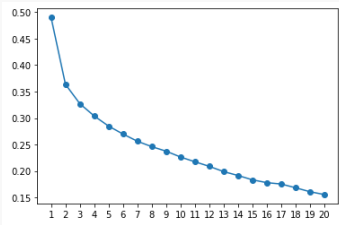
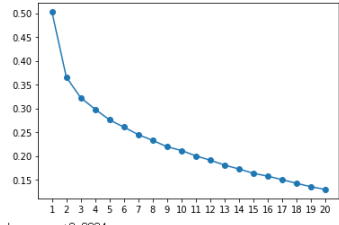
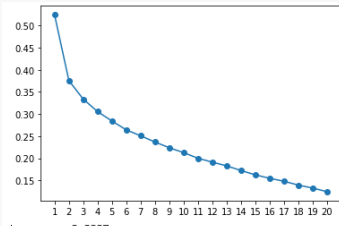
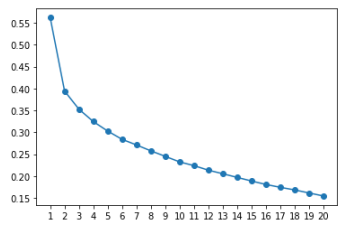
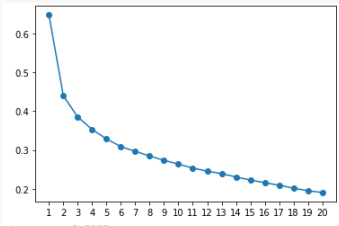
로 결과가 나옵니다.

각각의 다른 loss function을 적용시켰기 때문에, training loss의 비교는 모델의 학습정도와 상관관계가 없기 때문에 accuracy만 비교해보면, CrossEntropyLoss가 가장 높은 accuracy를 보이며, 그다음으로는 MultiMarginLoss, 마지막으로 MSELoss 가 가장 낮은 accuracy를 보였습니다.

4-2-3 LeNet-5 batch size

`batch_size=100,`

기본적으로 설정되어 있는 batch size는 100이며, batchsize를 조정해가며 학습 양상을 관찰해봤습니다.

<p>using device : cuda training epoch : 20 batch size : 10 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.490047864468175 epoch : 2 training_loss : 0.3636420541096784 epoch : 3 training_loss : 0.32711836575219905 epoch : 4 training_loss : 0.3038830161086911 epoch : 5 training_loss : 0.28476807569762664 epoch : 6 training_loss : 0.269235923978687 epoch : 7 training_loss : 0.25617936368393226 epoch : 8 training_loss : 0.2459706564038788 epoch : 9 training_loss : 0.2374365976064009 epoch : 10 training_loss : 0.22639539529454405 epoch : 11 training_loss : 0.21741124068177295 epoch : 12 training_loss : 0.20860081386576507 epoch : 13 training_loss : 0.19879989888141206 epoch : 14 training_loss : 0.19151388682441805 epoch : 15 training_loss : 0.18330630474250992 epoch : 16 training_loss : 0.1779866018563043 epoch : 17 training_loss : 0.1754032856208847 epoch : 18 training_loss : 0.16831223414594854 epoch : 19 training_loss : 0.16063526366096225 epoch : 20 training_loss : 0.15553122502251243 </pre>	 <p>Accuracy : 0.8819</p>
<p>using device : cuda training epoch : 20 batch size : 20 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.502605277194332 epoch : 2 training_loss : 0.3654025566533694 epoch : 3 training_loss : 0.3223018933019294 epoch : 4 training_loss : 0.2982845311177273 epoch : 5 training_loss : 0.2757666798954 epoch : 6 training_loss : 0.26056827254655385 epoch : 7 training_loss : 0.24485771462445427 epoch : 8 training_loss : 0.23269280381485233 epoch : 9 training_loss : 0.21934733038147275 epoch : 10 training_loss : 0.21148178397227665 epoch : 11 training_loss : 0.20016694400536 epoch : 12 training_loss : 0.1911335817448951 epoch : 13 training_loss : 0.18076299664302414 epoch : 14 training_loss : 0.17252839203604836 epoch : 15 training_loss : 0.1636033520294205 epoch : 16 training_loss : 0.1574541158303205 epoch : 17 training_loss : 0.15015844544629565 epoch : 18 training_loss : 0.1425333518046728 epoch : 19 training_loss : 0.13527627422571353 epoch : 20 training_loss : 0.1295689219551471 </pre>	 <p>Accuracy : 0.8894</p>
<p>using device : cuda training epoch : 20 batch size : 40 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.5243840074588206 epoch : 2 training_loss : 0.375480889260769 epoch : 3 training_loss : 0.33357662116984527 epoch : 4 training_loss : 0.3054156487161915 epoch : 5 training_loss : 0.2841361474494136 epoch : 6 training_loss : 0.2641371151308219 epoch : 7 training_loss : 0.25065300916135325 epoch : 8 training_loss : 0.23663412949194565 epoch : 9 training_loss : 0.22365360959833683 epoch : 10 training_loss : 0.21257398336884023 epoch : 11 training_loss : 0.19965433948412581 epoch : 12 training_loss : 0.19105938033014527 epoch : 13 training_loss : 0.18260704667617852 epoch : 14 training_loss : 0.17218717679505882 epoch : 15 training_loss : 0.1622757335950933 epoch : 16 training_loss : 0.1545366797055716 epoch : 17 training_loss : 0.14778970481206977 epoch : 18 training_loss : 0.1391778212382148 epoch : 19 training_loss : 0.1324356447135623 epoch : 20 training_loss : 0.12426875157840566 </pre>	 <p>Accuracy : 0.8897</p>
<p>using device : cuda training epoch : 20 batch size : 80 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.562104420363903 epoch : 2 training_loss : 0.39429932395617145 epoch : 3 training_loss : 0.3527688120206197 epoch : 4 training_loss : 0.32453971490263966 epoch : 5 training_loss : 0.30281769379973406 epoch : 6 training_loss : 0.28399891965587953 epoch : 7 training_loss : 0.2711497734884423 epoch : 8 training_loss : 0.25772222065925604 epoch : 9 training_loss : 0.2451436433692775 epoch : 10 training_loss : 0.23242941287159824 epoch : 11 training_loss : 0.2239504806300006 epoch : 12 training_loss : 0.21354027469456185 epoch : 13 training_loss : 0.20547138496736705 epoch : 14 training_loss : 0.1970672227193912 epoch : 15 training_loss : 0.18883771195703368 epoch : 16 training_loss : 0.18092950917780412 epoch : 17 training_loss : 0.17440517259140803 epoch : 18 training_loss : 0.1684788446350898 epoch : 19 training_loss : 0.16150352256496744 epoch : 20 training_loss : 0.15463724631567802 </pre>	 <p>Accuracy : 0.8948</p>
<p>using device : cuda training epoch : 20 batch size : 160 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name: Adam loss function : CrossEntropyLoss activation function: tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</p>	<pre> epoch : 1 training_loss : 0.6476017827192942 epoch : 2 training_loss : 0.43956394004821786 epoch : 3 training_loss : 0.3849507731199266 epoch : 4 training_loss : 0.3530229199727376 epoch : 5 training_loss : 0.32842888263861353 epoch : 6 training_loss : 0.3089893394027482 epoch : 7 training_loss : 0.2965893056720097 epoch : 8 training_loss : 0.28474243024985013 epoch : 9 training_loss : 0.2735812323792774 epoch : 10 training_loss : 0.26368495782216383 epoch : 11 training_loss : 0.25368911663691207 epoch : 12 training_loss : 0.24562547882397961 epoch : 13 training_loss : 0.23932462889556063 epoch : 14 training_loss : 0.23073950258890785 epoch : 15 training_loss : 0.22276596411069255 epoch : 16 training_loss : 0.21587232472521477 epoch : 17 training_loss : 0.2095828085939089 epoch : 18 training_loss : 0.20181408460934958 epoch : 19 training_loss : 0.19522839909791953 epoch : 20 training_loss : 0.1907044083078702 </pre>	 <p>Accuracy : 0.8956</p>

<pre> using device : cuda training epoch : 20 batch size : 10 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.490047864468175 epoch : 2 training_loss : 0.36364205410996784 epoch : 3 training_loss : 0.32711836575219905 epoch : 4 training_loss : 0.3038830161086911 epoch : 5 training_loss : 0.2847687959762664 epoch : 6 training_loss : 0.2696235923978687 epoch : 7 training_loss : 0.2561793636838226 epoch : 8 training_loss : 0.2459706564038788 epoch : 9 training_loss : 0.2374365976064009 epoch : 10 training_loss : 0.22639339529454405 epoch : 11 training_loss : 0.21741124068177295 epoch : 12 training_loss : 0.2080081385776507 epoch : 13 training_loss : 0.1987998988141206 epoch : 14 training_loss : 0.19151388682441805 epoch : 15 training_loss : 0.18330830474250992 epoch : 16 training_loss : 0.1779866018583043 epoch : 17 training_loss : 0.175403286208847 epoch : 18 training_loss : 0.16831223414594854 epoch : 19 training_loss : 0.16083526368096225 epoch : 20 training_loss : 0.15553122502251243 </pre>	<p>Accuracy : 0.8819</p>
<pre> using device : cuda training epoch : 20 batch size : 320 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.7840733944094758 epoch : 2 training_loss : 0.5106830529889136 epoch : 3 training_loss : 0.4402470961207993 epoch : 4 training_loss : 0.401993964363027 epoch : 5 training_loss : 0.3736253242122935 epoch : 6 training_loss : 0.35549239240546826 epoch : 7 training_loss : 0.340960682554041 epoch : 8 training_loss : 0.32775931259527546 epoch : 9 training_loss : 0.31412057929179255 epoch : 10 training_loss : 0.30727584014601894 epoch : 11 training_loss : 0.29810036591348815 epoch : 12 training_loss : 0.2883669225952866 epoch : 13 training_loss : 0.2804493785382593 epoch : 14 training_loss : 0.27089761747395824 epoch : 15 training_loss : 0.26563294399230863 epoch : 16 training_loss : 0.2589552732393703 epoch : 17 training_loss : 0.2525888067607523 epoch : 18 training_loss : 0.24969614265198768 epoch : 19 training_loss : 0.24312586996305363 epoch : 20 training_loss : 0.23478551376312168 </pre>	<p>Accuracy : 0.8877</p>
<pre> using device : cuda training epoch : 20 batch size : 640 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 0.9140648059947515 epoch : 2 training_loss : 0.5487910730223501 epoch : 3 training_loss : 0.4825555627704948 epoch : 4 training_loss : 0.4413999376759451 epoch : 5 training_loss : 0.4097939565485766 epoch : 6 training_loss : 0.3917608126517264 epoch : 7 training_loss : 0.37249942428322236 epoch : 8 training_loss : 0.3596828922789583 epoch : 9 training_loss : 0.3459203121482686 epoch : 10 training_loss : 0.33592101675207897 epoch : 11 training_loss : 0.32751354671293686 epoch : 12 training_loss : 0.3182812624721118 epoch : 13 training_loss : 0.3119272435665095 epoch : 14 training_loss : 0.3071766716818656 epoch : 15 training_loss : 0.29614500092562795 epoch : 16 training_loss : 0.29064219436978767 epoch : 17 training_loss : 0.2840363295168004 epoch : 18 training_loss : 0.2824815728010669 epoch : 19 training_loss : 0.27534176857881654 epoch : 20 training_loss : 0.2709989164785671 </pre>	<p>Accuracy : 0.8824</p>
<pre> using device : cuda training epoch : 20 batch size : 1280 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False </pre>	<pre> epoch : 1 training_loss : 1.06914623664773 epoch : 2 training_loss : 0.6275417234598632 epoch : 3 training_loss : 0.5457656409429466 epoch : 4 training_loss : 0.49703644475211267 epoch : 5 training_loss : 0.4639795444498891 epoch : 6 training_loss : 0.43823557962541987 epoch : 7 training_loss : 0.4198289610769438 epoch : 8 training_loss : 0.4040280029825542 epoch : 9 training_loss : 0.38603574426277615 epoch : 10 training_loss : 0.3785465897425362 epoch : 11 training_loss : 0.3697747367879619 epoch : 12 training_loss : 0.36109061927989067 epoch : 13 training_loss : 0.35442300270433014 epoch : 14 training_loss : 0.34497399045073474 epoch : 15 training_loss : 0.33721985246824177 epoch : 16 training_loss : 0.33163866076790407 epoch : 17 training_loss : 0.3252354018068241 epoch : 18 training_loss : 0.3212984834919806 epoch : 19 training_loss : 0.31746046180310444 epoch : 20 training_loss : 0.31036426710045856 </pre>	<p>Accuracy : 0.8729</p>

batch size를 10에서부터 1280까지 2배씩 증가시키며 학습을 해보았고, 결과를 순차적으로 나열해 보면

- batch size : 10 -> 최종 training loss : 0.1555, accuracy : 88.19%
- batch size : 20 -> 최종 training loss : 0.1295, accuracy : 88.94%
- batch size : 40 -> 최종 training loss : 0.1242, accuracy : 88.97%
- batch size : 80 -> 최종 training loss : 0.1546, accuracy : 89.48%
- batch size : 160 -> 최종 training loss : 0.1907, accuracy : 89.56%
- batch size : 320 -> 최종 training loss : 0.2347, accuracy : 88.77%
- batch size : 640 -> 최종 training loss : 0.2709, accuracy : 88.24%
- batch size : 1280 -> 최종 training loss : 0.3103, accuracy : 87.29%

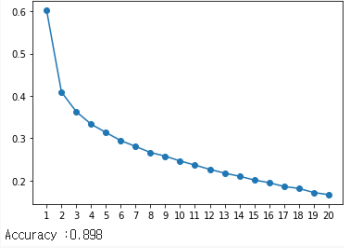
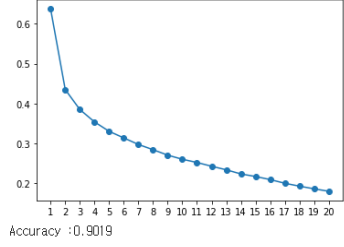
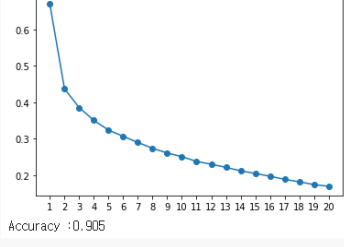
로 결과가 나옵니다. batch size를 증가시킬수록 최종 training loss 는 커지는 양상을 보이고, accuracy 는 올라갔다가 내려가는 양상이 나타났습니다.

4-2-4 LeNet-5 be creative and analytical

activation function

```
activation_function='tanh',
```

기본적으로 tanh 함수로 설정되어 있고, relu, leaky relu를 사용할 수 있습니다.

<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : tanh convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 0.6022232569091547 epoch : 2 training_loss : 0.4093605014433464 epoch : 3 training_loss : 0.36291954206923677 epoch : 4 training_loss : 0.33346481939156847 epoch : 5 training_loss : 0.3136953362077473 epoch : 6 training_loss : 0.2945469029254348 epoch : 7 training_loss : 0.28085130119075397 epoch : 8 training_loss : 0.2665092867486663 epoch : 9 training_loss : 0.2578994916503625 epoch : 10 training_loss : 0.2464731881270805 epoch : 11 training_loss : 0.2367818635329602 epoch : 12 training_loss : 0.22663164985676598 epoch : 13 training_loss : 0.21787621052314848 epoch : 14 training_loss : 0.21039037581533201 epoch : 15 training_loss : 0.20152541413903224 epoch : 16 training_loss : 0.19516785579423107 epoch : 17 training_loss : 0.18630203723907468 epoch : 18 training_loss : 0.18166960794478643 epoch : 19 training_loss : 0.1719982453187305 epoch : 20 training_loss : 0.16686947919428383</pre>	 <p>Accuracy : 0.898</p>
<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : relu convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 0.6376726399362093 epoch : 2 training_loss : 0.43507320451239767 epoch : 3 training_loss : 0.384996630847059 epoch : 4 training_loss : 0.353741263796866 epoch : 5 training_loss : 0.33006159608562785 epoch : 6 training_loss : 0.31355175179739925 epoch : 7 training_loss : 0.29708753209561134 epoch : 8 training_loss : 0.2839416490991908 epoch : 9 training_loss : 0.27029314730316406 epoch : 10 training_loss : 0.25994927664597844 epoch : 11 training_loss : 0.2518402079865337 epoch : 12 training_loss : 0.24234253451228122 epoch : 13 training_loss : 0.2334366928289336 epoch : 14 training_loss : 0.2230776788915198 epoch : 15 training_loss : 0.21658677354454975 epoch : 16 training_loss : 0.2090969909727578 epoch : 17 training_loss : 0.19966113653039965 epoch : 18 training_loss : 0.19232454987242834 epoch : 19 training_loss : 0.18571045115590112 epoch : 20 training_loss : 0.179511470731596</pre>	 <p>Accuracy : 0.9019</p>
<pre>using device : cuda training epoch : 20 batch size : 100 learning rate : 0.001 scheduler name : None scheduler step size : None scheduler gamma : None optimizer name : Adam loss function : CrossEntropyLoss activation function : leaky_relu convolutional channels : 6 16 120 using batch normalization? : False pooling layer : avg using dropout layer : False</pre>	<pre>epoch : 1 training_loss : 0.6696104001998897 epoch : 2 training_loss : 0.437105017080903 epoch : 3 training_loss : 0.38510479847590146 epoch : 4 training_loss : 0.36020899434884395 epoch : 5 training_loss : 0.3237428560107945 epoch : 6 training_loss : 0.30635937766482413 epoch : 7 training_loss : 0.2895328145225844 epoch : 8 training_loss : 0.27364528931677323 epoch : 9 training_loss : 0.2606726797918479 epoch : 10 training_loss : 0.25058821669469244 epoch : 11 training_loss : 0.23747536748647688 epoch : 12 training_loss : 0.23002479876702053 epoch : 13 training_loss : 0.22113121223325552 epoch : 14 training_loss : 0.21173419078812 epoch : 15 training_loss : 0.2044892825062076 epoch : 16 training_loss : 0.19592093872775665 epoch : 17 training_loss : 0.18837734101961046 epoch : 18 training_loss : 0.18154624622936036 epoch : 19 training_loss : 0.17388720681890837 epoch : 20 training_loss : 0.16929311466092886</pre>	 <p>Accuracy : 0.905</p>

순차적으로 activation function으로 tanh, relu, leaky relu 함수를 써봤고, 결과를 나열해 보면,

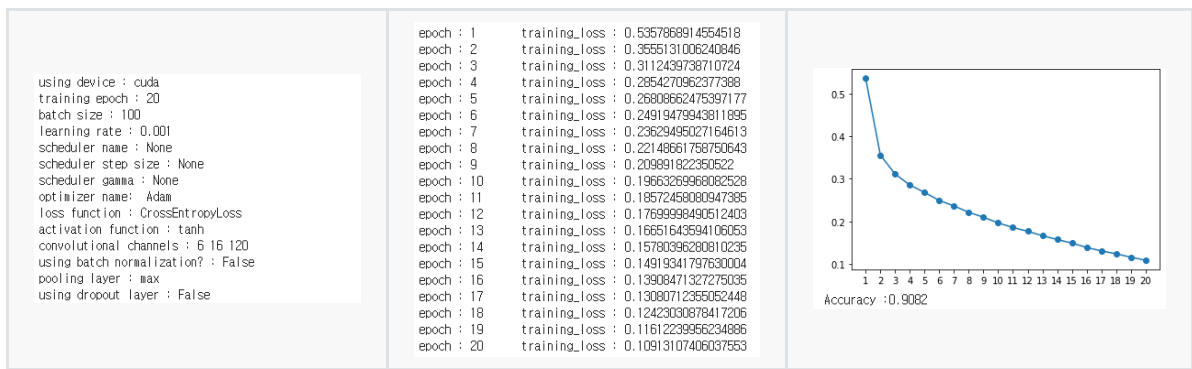
- tanh function -> 최종 training loss : 0.1668, accuracy : 89.8%
- relu function -> 최종 training loss : 0.1795, accuracy : 90.19%
- leaky relu function -> 최종 training loss : 0.1692, accuracy : 90.5 %

로 결과가 나옵니다. leaky relu에서 가장 높은 accuracy가 나왔고, 그 다음으로 relu 에서, 그리고 마지막으로 tanh 에서 가장 낮은 accuracy가 나왔습니다.

pooling layer

```
pooling_layer='avg',
```

기본적으로 average pooling layer로 설정되어 있고, max pooling layer('max') 로 설정 할 수 있습니다.

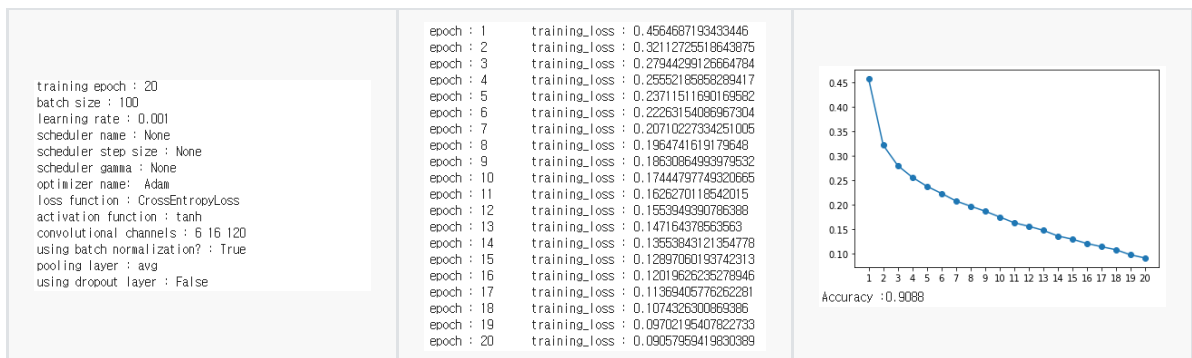


max pooling layer를 적용해본 결과, accuracy는 90.82%로 기존의 average pooling layer를 사용한 base case에 비해 높게 나왔습니다.

batch normalization

`use_batchnorm=False,`

기본적으로 batch normalization layer를 사용하지 않고, 설정으로 True로 넘겨주면 batch normalization layer가 추가되게 됩니다.

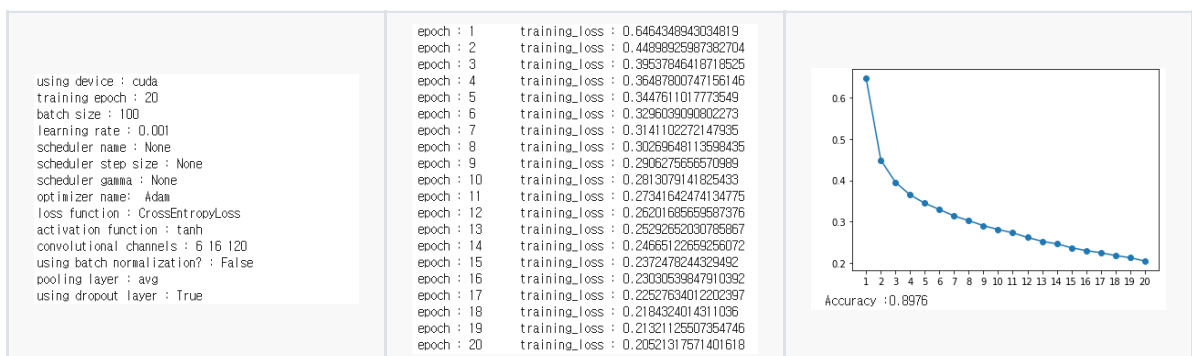


batch normalization을 추가해본 결과, base case의 accuracy (89.24%) 에 비해 향상된 accuracy, 90.88%가 관측되었습니다.

dropout

`use_dropout=False):`

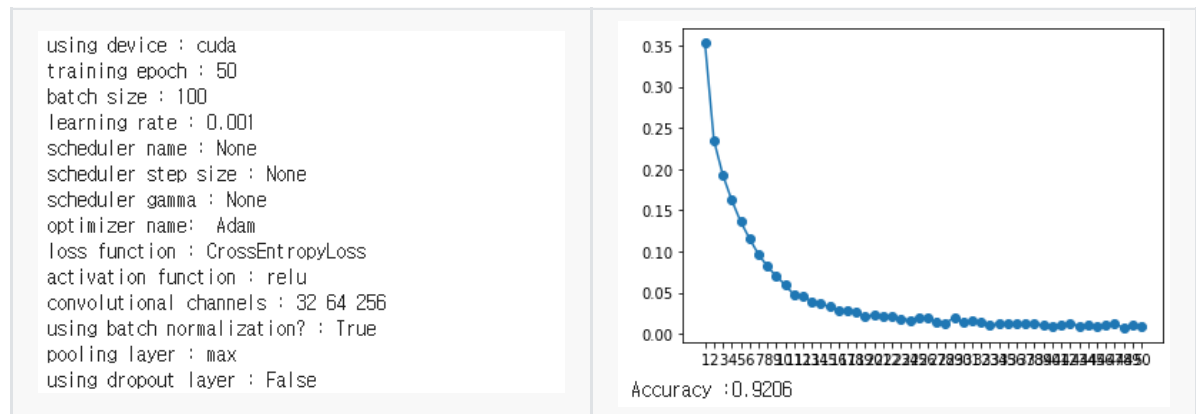
기본적으로 dropout 기법을 사용하지 않으며, 설정으로 True를 넘겨주면 첫번째 fully connected layer에 dropout을 적용시킬 수 있게 하였습니다.



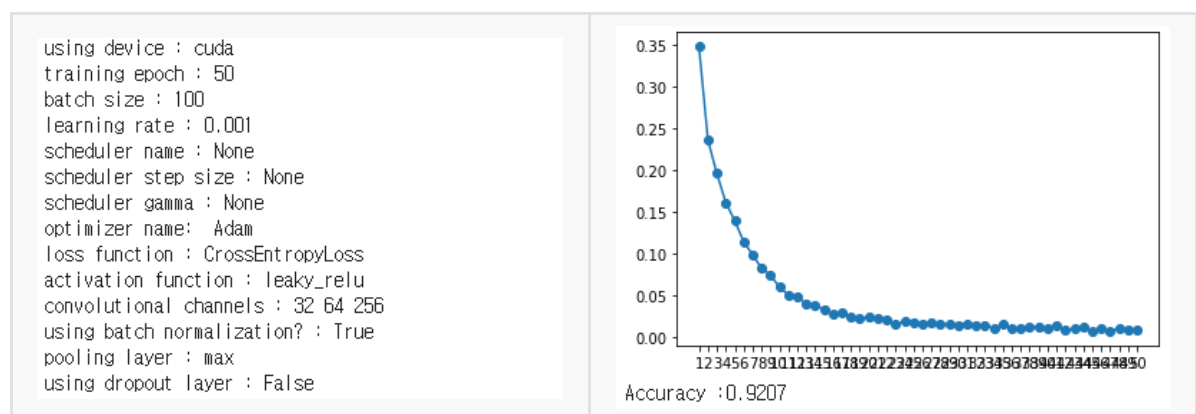
추가해본 결과, base case에비해서 accuracy는 89.76%로 향상된 결과가 나왔으며, 최종 training loss는 기존에 비해 0.2052로 더 높게 나왔습니다.

Setting many parameters at once

위에서 parameter를 하나씩 조정해본 결과들을 참고해서, 결과가 잘 나온 parameter들을 동시에 한번에 조정해서 학습정도를 관찰해봤습니다.



위와 같이 convolutional channel size를 각각 32, 64, 256으로 주고 max pooling layer를 사용, batch normalization layer를 사용, 그리고 relu를 activation function으로 사용한 결과 accuracy가 92.06%까지 나왔고, training loss는 0.0098까지 떨어졌습니다.



위와 같이 설정한 상태에서는 92.07%까지 accuracy가 나왔고, training loss는 0.0096떨어졌습니다.

5. Difference between Fully connected Network and LeNet-5

Fully connected network에서는 fully connected layer로만 연결되어 있기 때문에 overfitting이 쉽게 일어날 수 있다는 단점이 있고, image의 spatial feature에 대한 abstraction을 하지 않는다.

LeNet-5과 같은 경우 LeNet-5가 처음으로 소개된 경우의 모델을 보면, convolutional layer를 통해서 spatial feature에대한 abstraction도 진행하고, average pooling layer를 통해서 data의 크기도 줄이지만, convolutional filter에서 channel size가 충분하지 않고, 마지막 두개의 fully connected layer에서 dropout 기법을 적용시키지 않아 overfitting될 가능성이 있어 fully connected network에 비해

accuracy가 낮게 나왔습니다 (89.24% < 89.4%).

하지만 LeNet-5에서 convolutional channel size를 조정하고, activation function을 leaky relu function으로 바꾸고, average pooling layer 대신 max pooling layer 를 사용하고, batch normalization을 진행한 결과 92.07%까지 accuracy를 올릴 수 있었습니다.

6. Analysis of effects of optimizer parameters, loss function, batch size, activation function, pooling layer, convolutional channel size, batch normalization, dropout layer

optimizer parameter - scheduler

scheduler 같은 경우에는 특정 몇개의 data를 통과할때마다 learning rate에 상수를 곱해주는 StepLR scheduler를 사용해 봤습니다. scheduler를 적용시켜본 결과, learning rate를 너무 빨리 줄일 경우, learning rate가 너무 작아져서 training loss가 안줄어드는 현상이 발생했으며, learning rate 를 너무 안 줄인 경우, 학습이 잘 되지 않는 현상이 발생했습니다.

scheduler 같은 경우 현재 training loss가 어떠한 방식으로 줄어드는지를 염두에 안두고, 몇개의 data가 지나갈때마다 learning rate에 상수를 곱하는 방식으로 learning rate를 조정하기 때문에 training loss 는 더 낮게 나오는 경우는 있어도, accuracy가 눈에 띄는 정도로 올라가지는 않았습니다. 좀더 세밀하게 조정하기 위해서는 현재 training loss를 추적하면서, loss가 갑자기 내려가는 정도가 낮아질 경우, learning rate를 조정하는 방식이 accuracy 향상에 더 효율적일 것 같습니다.

optimizer parameter - learning rate

learning rate 같은 경우 너무 낮은 경우, training loss가 잘 안줄어들고, 학습이 진전이 잘 안되었습니다. learning rate가 너무 큰 경우, loss function에서의 minimum 값을 잘 찾지 못해 training loss가 다음 epoch에서 증가하는 현상까지 일어났습니다.

여러가지 learning rate로 실험해본 결과, 0.0008일때 accuracy가 89.39%로 가장 높게 나왔습니다.

optimizer parameter - optimizer type

optimizer type 같은 경우 SGD, RMSProp, Adam optimizer를 사용해봤고, SGD optimizer 같은 경우에는 high computation의 문제와 saddle point에 대해서 벗어나지 못할 가능성이 있습니다, 이러한 문제 점을 해결하기 위해 순차적으로 나온 개선 optimizer가 RMSProp, Adam optimizer고, 예상한 바와 같이 순서대로 SGD는 가장 낮은 accuracy를, Adam는 가장 높은 accuracy를 나타냈습니다.

loss function

loss function 은 MSELoss, MultiMarginLoss, CrossEntropyLoss를 사용해봤습니다. MSELoss 같은 경우 target class에 해당하는 one hot encoding tensor와 model output data의 softmax tensor 의 거리의 제곱만을 고려했기 때문에 다른 loss 들에 비해 낮은 accuracy가 나왔습니다. MultiMarginLoss 같은 경우, input 과 target의 뺄값이 margin보다 낮은 경우 loss를 0으로 취급하기 때문에 학습 결과 CrossEntropyLoss에 비해 낮은 accuracy가 나왔습니다. 마지막으로 CrossEntropyLoss 같은 경우에 probability distribution에 관한 distance를 측정하기 때문에 가장 높은 accuracy가 나왔습니다.

batch size

batch size 가 너무 작은 경우에, 일단 모델을 학습시키는데에 오래 걸리고, 너무 small batch에 대해서 학습을 진행하기 때문에 sample bias가 존재합니다. 하지만 large batch를 사용할 경우, overfitting 이 더 잘 발생합니다. 이러한 점에서 적당한 batch size를 이용하는게 좋으며, 여러가지 실험해본 결과 160 일때 최고의 accuracy가 나왔습니다.

activation function

activation function으로는 tanh, relu, leaky relu function을 사용해봤습니다. tanh function 같은 경우 양단에서 기울기가 급격하게 0으로 수렴하는 현상 때문에 gradient descent에서 문제점이 발생합니다. relu function 같은 경우 값이 0인 구간으로 인해 weight initialization을 잘못 할 경우, 평생 출력값이 0 일수 있는 문제점이 발생합니다. 이러한 문제점을 해결하기 위해 결과값이 0인 범위에 대해 작은 기울기를 준 함수가 leaky relu 입니다. 차례대로 개선된 만큼 tanh, relu, leaky relu 순으로 높은 accuracy가 나왔습니다.

pooling layer

pooling layer 같은 경우 average와 max pooling layer를 사용해봤으며, max pooling layer에서 더 높은 accuracy가 나왔습니다. 이러한 이유는 down sampling하는 과정에서 평균을 구하는것보다 space를 대표하는 가장 큰 값을 뽑는게 더 좋기 때문입니다.

convolutional channel size

convolutional channel size는 기존에 6,16,120 이외에 32, 64, 256으로 channel size를 조정해 보았고, 더 높은 accuracy를 보였습니다. 높은 channel 갯수를 통해서 더 많은 feature들에 대한 학습을 진행할 수 있어서 더 높은 accuracy가 나왔습니다.

batch normalization

layer에서 layer로 전해지는 값들이 너무 크거나 작을 경우, gradient descent를 할때 문제가 생깁니다. 이를 방지하기 위해서 batch normalization을 통해 값들을 조정할 경우 더 높은 accuracy가 나왔습니다.

dropout layer

fully connected layer 같은 경우 weight가 너무 많아서 training data에서만 학습이 잘 되는 현상인 overfitting의 문제점이 일어날 수 있습니다. 이를 위해서 drop out 기법을 통해서 일정확률로 neuron의 output data를 0으로 만들어주는 dropout layer를 추가하면 overfitting 현상을 해결할 수 있습니다.

