auto encoder

# example code

|  |
| --- |
| ''' 1. Module Import '''  import numpy as np  import matplotlib.pyplot as plt  import torch  import torch.nn as nn  import torch.nn.functional as F  from torchvision import transforms, datasets  ''' 2. 딥러닝 모델을 설계할 때 활용하는 장비 확인 '''  if torch.cuda.is\_available():      DEVICE = torch.device('cuda')  else:      DEVICE = torch.device('cpu')  print('Using PyTorch version:', torch.\_\_version\_\_, ' Device:', DEVICE)  BATCH\_SIZE = 32  EPOCHS = 10  ''' 3. FashionMNIST 데이터 다운로드 (Train set, Test set 분리하기) '''  train\_dataset = datasets.FashionMNIST(root = "../data/FashionMNIST",                                        train = True,                                        download = True,                                        transform = transforms.ToTensor())  test\_dataset = datasets.FashionMNIST(root = "../data/FashionMNIST",                                       train = False,                                       transform = transforms.ToTensor())  train\_loader = torch.utils.data.DataLoader(dataset = train\_dataset,                                             batch\_size = BATCH\_SIZE,                                             shuffle = True)  test\_loader = torch.utils.data.DataLoader(dataset = test\_dataset,                                            batch\_size = BATCH\_SIZE,                                            shuffle = False)  ''' 4. 데이터 확인하기 (1) '''  for (X\_train, y\_train) in train\_loader:      print('X\_train:', X\_train.size(), 'type:', X\_train.type())      print('y\_train:', y\_train.size(), 'type:', y\_train.type())      break  ''' 5. 데이터 확인하기 (2) '''  pltsize = 1  plt.figure(figsize=(10 \* pltsize, pltsize))  for i in range(10):      plt.subplot(1, 10, i + 1)      plt.axis('off')      plt.imshow(X\_train[i, :, :, :].numpy().reshape(28, 28), cmap = "gray\_r")      plt.title('Class: ' + str(y\_train[i].item()))  ''' 6. AutoEncoder (AE) 모델 설계하기 '''  class AE(nn.Module):      def \_\_init\_\_(self):          super(AE, self).\_\_init\_\_()            self.encoder = nn.Sequential(              nn.Linear(28 \* 28, 512),              nn.ReLU(),              nn.Linear(512, 256),              nn.ReLU(),              nn.Linear(256, 32),)            self.decoder = nn.Sequential(              nn.Linear(32, 256),              nn.ReLU(),              nn.Linear(256, 512),              nn.ReLU(),              nn.Linear(512, 28 \* 28),)      def forward(self, x):          encoded = self.encoder(x)          decoded = self.decoder(encoded)          return encoded, decoded  ''' 7. Optimizer, Objective Function 설정하기 '''  model = AE().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)  criterion = nn.MSELoss()  print(model)  ''' 8. AE 모델 학습을 진행하며 학습 데이터에 대한 모델 성능을 확인하는 함수 정의 '''  def train(model, train\_loader, optimizer, log\_interval):      model.train()      for batch\_idx, (image, \_) in enumerate(train\_loader):          image = image.view(-1, 28 \* 28).to(DEVICE)          target = image.view(-1, 28 \* 28).to(DEVICE)          optimizer.zero\_grad()          encoded, decoded = model(image)          loss = criterion(decoded, target)          loss.backward()          optimizer.step()          if batch\_idx % log\_interval == 0:              print("Train Epoch: {} [{}/{} ({:.0f}%)]\tTrain Loss: {:.6f}".format(                  epoch, batch\_idx \* len(image),                  len(train\_loader.dataset), 100. \* batch\_idx / len(train\_loader),                  loss.item()))  ''' 9. 학습되는 과정 속에서 검증 데이터에 대한 모델 성능을 확인하는 함수 정의 '''  def evaluate(model, test\_loader):      model.eval()      test\_loss = 0      real\_image = []      gen\_image = []      with torch.no\_grad():          for image, \_ in test\_loader:              image = image.view(-1, 28 \* 28).to(DEVICE)              target = image.view(-1, 28 \* 28).to(DEVICE)              encoded, decoded = model(image)                test\_loss += criterion(decoded, image).item()              real\_image.append(image.to("cpu"))              gen\_image.append(decoded.to("cpu"))        test\_loss /= (len(test\_loader.dataset) / BATCH\_SIZE)      return test\_loss, real\_image, gen\_image  ''' 10. AutoEncoder 학습 실행하며 Test set의 Reconstruction Error 확인하기 '''  for epoch in range(1, EPOCHS + 1):      train(model, train\_loader, optimizer, log\_interval = 200)      test\_loss, real\_image, gen\_image = evaluate(model, test\_loader)      print("\n[EPOCH: {}], \tTest Loss: {:.4f}".format(epoch, test\_loss))      f, a = plt.subplots(2, 10, figsize = (10, 4))      for i in range(10):          img = np.reshape(real\_image[0][i], (28, 28))          a[0][i].imshow(img, cmap = "gray\_r")          a[0][i].set\_xticks(())          a[0][i].set\_yticks(())        for i in range(10):          img = np.reshape(gen\_image[0][i], (28, 28))          a[1][i].imshow(img, cmap = "gray\_r")          a[1][i].set\_xticks(())          a[1][i].set\_yticks(())      plt.show() |

# testing result

|  |
| --- |
| Train Epoch: 1 [0/60000 (0%)] Train Loss: 0.209500  Train Epoch: 1 [6400/60000 (11%)] Train Loss: 0.026070  Train Epoch: 1 [12800/60000 (21%)] Train Loss: 0.021709  Train Epoch: 1 [19200/60000 (32%)] Train Loss: 0.018289  Train Epoch: 1 [25600/60000 (43%)] Train Loss: 0.019566  Train Epoch: 1 [32000/60000 (53%)] Train Loss: 0.019107  Train Epoch: 1 [38400/60000 (64%)] Train Loss: 0.017756  Train Epoch: 1 [44800/60000 (75%)] Train Loss: 0.015910  Train Epoch: 1 [51200/60000 (85%)] Train Loss: 0.018428  Train Epoch: 1 [57600/60000 (96%)] Train Loss: 0.015730  [EPOCH: 1], Test Loss: 0.0151  Train Epoch: 2 [0/60000 (0%)] Train Loss: 0.017791  Train Epoch: 2 [6400/60000 (11%)] Train Loss: 0.012933  Train Epoch: 2 [12800/60000 (21%)] Train Loss: 0.013861  Train Epoch: 2 [19200/60000 (32%)] Train Loss: 0.014870  Train Epoch: 2 [25600/60000 (43%)] Train Loss: 0.013782  Train Epoch: 2 [32000/60000 (53%)] Train Loss: 0.013975  Train Epoch: 2 [38400/60000 (64%)] Train Loss: 0.015480  Train Epoch: 2 [44800/60000 (75%)] Train Loss: 0.015584  Train Epoch: 2 [51200/60000 (85%)] Train Loss: 0.010608  Train Epoch: 2 [57600/60000 (96%)] Train Loss: 0.014494  [EPOCH: 2], Test Loss: 0.0129  Train Epoch: 3 [0/60000 (0%)] Train Loss: 0.012810  Train Epoch: 3 [6400/60000 (11%)] Train Loss: 0.014110  Train Epoch: 3 [12800/60000 (21%)] Train Loss: 0.010185  Train Epoch: 3 [19200/60000 (32%)] Train Loss: 0.011491  Train Epoch: 3 [25600/60000 (43%)] Train Loss: 0.011818  Train Epoch: 3 [32000/60000 (53%)] Train Loss: 0.012162  Train Epoch: 3 [38400/60000 (64%)] Train Loss: 0.015404  Train Epoch: 3 [44800/60000 (75%)] Train Loss: 0.011720  Train Epoch: 3 [51200/60000 (85%)] Train Loss: 0.010931  Train Epoch: 3 [57600/60000 (96%)] Train Loss: 0.010638  [EPOCH: 3], Test Loss: 0.0120  Train Epoch: 4 [0/60000 (0%)] Train Loss: 0.010915  Train Epoch: 4 [6400/60000 (11%)] Train Loss: 0.014795  Train Epoch: 4 [12800/60000 (21%)] Train Loss: 0.010404  Train Epoch: 4 [19200/60000 (32%)] Train Loss: 0.009930  Train Epoch: 4 [25600/60000 (43%)] Train Loss: 0.011077  Train Epoch: 4 [32000/60000 (53%)] Train Loss: 0.010136  Train Epoch: 4 [38400/60000 (64%)] Train Loss: 0.010642  Train Epoch: 4 [44800/60000 (75%)] Train Loss: 0.010750  Train Epoch: 4 [51200/60000 (85%)] Train Loss: 0.011340  Train Epoch: 4 [57600/60000 (96%)] Train Loss: 0.012551  [EPOCH: 4], Test Loss: 0.0115  Train Epoch: 5 [0/60000 (0%)] Train Loss: 0.010041  Train Epoch: 5 [6400/60000 (11%)] Train Loss: 0.009946  Train Epoch: 5 [12800/60000 (21%)] Train Loss: 0.012546  Train Epoch: 5 [19200/60000 (32%)] Train Loss: 0.010338  Train Epoch: 5 [25600/60000 (43%)] Train Loss: 0.010648  Train Epoch: 5 [32000/60000 (53%)] Train Loss: 0.012258  Train Epoch: 5 [38400/60000 (64%)] Train Loss: 0.009359  Train Epoch: 5 [44800/60000 (75%)] Train Loss: 0.009476  Train Epoch: 5 [51200/60000 (85%)] Train Loss: 0.010316  Train Epoch: 5 [57600/60000 (96%)] Train Loss: 0.010404  [EPOCH: 5], Test Loss: 0.0109  Train Epoch: 6 [0/60000 (0%)] Train Loss: 0.010356  Train Epoch: 6 [6400/60000 (11%)] Train Loss: 0.012726  Train Epoch: 6 [12800/60000 (21%)] Train Loss: 0.010679  Train Epoch: 6 [19200/60000 (32%)] Train Loss: 0.011042  Train Epoch: 6 [25600/60000 (43%)] Train Loss: 0.010268  Train Epoch: 6 [32000/60000 (53%)] Train Loss: 0.009594  Train Epoch: 6 [38400/60000 (64%)] Train Loss: 0.009111  Train Epoch: 6 [44800/60000 (75%)] Train Loss: 0.011282  Train Epoch: 6 [51200/60000 (85%)] Train Loss: 0.011089  Train Epoch: 6 [57600/60000 (96%)] Train Loss: 0.009392  [EPOCH: 6], Test Loss: 0.0105  Train Epoch: 7 [0/60000 (0%)] Train Loss: 0.009712  Train Epoch: 7 [6400/60000 (11%)] Train Loss: 0.011188  Train Epoch: 7 [12800/60000 (21%)] Train Loss: 0.010986  Train Epoch: 7 [19200/60000 (32%)] Train Loss: 0.009245  Train Epoch: 7 [25600/60000 (43%)] Train Loss: 0.011886  Train Epoch: 7 [32000/60000 (53%)] Train Loss: 0.008454  Train Epoch: 7 [38400/60000 (64%)] Train Loss: 0.010062  Train Epoch: 7 [44800/60000 (75%)] Train Loss: 0.009543  Train Epoch: 7 [51200/60000 (85%)] Train Loss: 0.010276  Train Epoch: 7 [57600/60000 (96%)] Train Loss: 0.012020  [EPOCH: 7], Test Loss: 0.0102  Train Epoch: 8 [0/60000 (0%)] Train Loss: 0.007712  Train Epoch: 8 [6400/60000 (11%)] Train Loss: 0.009814  Train Epoch: 8 [12800/60000 (21%)] Train Loss: 0.010263  Train Epoch: 8 [19200/60000 (32%)] Train Loss: 0.011020  Train Epoch: 8 [25600/60000 (43%)] Train Loss: 0.011255  Train Epoch: 8 [32000/60000 (53%)] Train Loss: 0.009000  Train Epoch: 8 [38400/60000 (64%)] Train Loss: 0.010220  Train Epoch: 8 [44800/60000 (75%)] Train Loss: 0.010973  Train Epoch: 8 [51200/60000 (85%)] Train Loss: 0.011835  Train Epoch: 8 [57600/60000 (96%)] Train Loss: 0.009824  [EPOCH: 8], Test Loss: 0.0100  Train Epoch: 9 [0/60000 (0%)] Train Loss: 0.008894  Train Epoch: 9 [6400/60000 (11%)] Train Loss: 0.010732  Train Epoch: 9 [12800/60000 (21%)] Train Loss: 0.010981  Train Epoch: 9 [19200/60000 (32%)] Train Loss: 0.008615  Train Epoch: 9 [25600/60000 (43%)] Train Loss: 0.009980  Train Epoch: 9 [32000/60000 (53%)] Train Loss: 0.010463  Train Epoch: 9 [38400/60000 (64%)] Train Loss: 0.012775  Train Epoch: 9 [44800/60000 (75%)] Train Loss: 0.009874  Train Epoch: 9 [51200/60000 (85%)] Train Loss: 0.008259  Train Epoch: 9 [57600/60000 (96%)] Train Loss: 0.010284  [EPOCH: 9], Test Loss: 0.0099  Train Epoch: 10 [0/60000 (0%)] Train Loss: 0.008283  Train Epoch: 10 [6400/60000 (11%)] Train Loss: 0.009601  Train Epoch: 10 [12800/60000 (21%)] Train Loss: 0.009874  Train Epoch: 10 [19200/60000 (32%)] Train Loss: 0.008390  Train Epoch: 10 [25600/60000 (43%)] Train Loss: 0.009748  Train Epoch: 10 [32000/60000 (53%)] Train Loss: 0.010440  Train Epoch: 10 [38400/60000 (64%)] Train Loss: 0.010192  Train Epoch: 10 [44800/60000 (75%)] Train Loss: 0.008686  Train Epoch: 10 [51200/60000 (85%)] Train Loss: 0.010374  Train Epoch: 10 [57600/60000 (96%)] Train Loss: 0.009301  [EPOCH: 10], Test Loss: 0.0098 |