CNN

# 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. CIFAR10 데이터 다운로드 (Train set, Test set 분리하기) '''  train\_dataset = datasets.CIFAR10(root = "../data/CIFAR\_10",                                    train = True,                                    download = True,                                    transform = transforms.ToTensor())  test\_dataset = datasets.CIFAR10(root = "../data/CIFAR\_10",                                  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(np.transpose(X\_train[i], (1, 2, 0)))      plt.title('Class: ' + str(y\_train[i].item()))  ''' 6. Convolutional Neural Network (CNN) 모델 설계하기 '''  class CNN(nn.Module):      def \_\_init\_\_(self):          super(CNN, self).\_\_init\_\_()          self.conv1 = nn.Conv2d(in\_channels = 3, out\_channels = 8, kernel\_size = 3, padding = 1)          self.conv2 = nn.Conv2d(in\_channels = 8, out\_channels = 16, kernel\_size = 3, padding = 1)          self.pool = nn.MaxPool2d(kernel\_size = 2, stride = 2)          self.fc1 = nn.Linear(8 \* 8 \* 16, 64)          self.fc2 = nn.Linear(64, 32)          self.fc3 = nn.Linear(32, 10)        def forward(self, x):          x = self.conv1(x)          x = F.relu(x)          x = self.pool(x)          x = self.conv2(x)          x = F.relu(x)          x = self.pool(x)            x = x.view(-1, 8 \* 8 \* 16)          x = self.fc1(x)          x = F.relu(x)          x = self.fc2(x)          x = F.relu(x)          x = self.fc3(x)          x = F.log\_softmax(x)          return x  ''' 7. Optimizer, Objective Function 설정하기 '''  model = CNN().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)  criterion = nn.CrossEntropyLoss()  print(model)  ''' 8. CNN 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, label) in enumerate(train\_loader):          image = image.to(DEVICE)          label = label.to(DEVICE)          optimizer.zero\_grad()          output = model(image)          loss = criterion(output, label)          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      correct = 0      with torch.no\_grad():          for image, label in test\_loader:              image = image.to(DEVICE)              label = label.to(DEVICE)              output = model(image)              test\_loss += criterion(output, label).item()              prediction = output.max(1, keepdim = True)[1]              correct += prediction.eq(label.view\_as(prediction)).sum().item()        test\_loss /= (len(test\_loader.dataset) / BATCH\_SIZE)      test\_accuracy = 100. \* correct / len(test\_loader.dataset)      return test\_loss, test\_accuracy  ''' 10. CNN 학습 실행하며 Train, Test set의 Loss 및 Test set Accuracy 확인하기 '''  for epoch in range(1, EPOCHS + 1):      train(model, train\_loader, optimizer, log\_interval = 200)      test\_loss, test\_accuracy = evaluate(model, test\_loader)      print("\n[EPOCH: {}], \tTest Loss: {:.4f}, \tTest Accuracy: {:.2f} % \n".format(          epoch, test\_loss, test\_accuracy)) |

# testing result

|  |
| --- |
| Train Epoch: 1 [0/50000 (0%)] Train Loss: 2.305517  Train Epoch: 1 [6400/50000 (13%)] Train Loss: 1.908107  Train Epoch: 1 [12800/50000 (26%)] Train Loss: 1.693745  Train Epoch: 1 [19200/50000 (38%)] Train Loss: 1.713397  Train Epoch: 1 [25600/50000 (51%)] Train Loss: 1.586022  Train Epoch: 1 [32000/50000 (64%)] Train Loss: 1.581453  Train Epoch: 1 [38400/50000 (77%)] Train Loss: 1.910545  Train Epoch: 1 [44800/50000 (90%)] Train Loss: 1.792278  [EPOCH: 1], Test Loss: 0.0472, Test Accuracy: 45.02 %  Train Epoch: 2 [0/50000 (0%)] Train Loss: 1.455860  Train Epoch: 2 [6400/50000 (13%)] Train Loss: 1.605248  Train Epoch: 2 [12800/50000 (26%)] Train Loss: 1.577104  Train Epoch: 2 [19200/50000 (38%)] Train Loss: 1.809724  Train Epoch: 2 [25600/50000 (51%)] Train Loss: 1.520014  Train Epoch: 2 [32000/50000 (64%)] Train Loss: 1.443045  Train Epoch: 2 [38400/50000 (77%)] Train Loss: 1.241889  Train Epoch: 2 [44800/50000 (90%)] Train Loss: 1.137973  [EPOCH: 2], Test Loss: 0.0414, Test Accuracy: 51.91 %  Train Epoch: 3 [0/50000 (0%)] Train Loss: 1.196696  Train Epoch: 3 [6400/50000 (13%)] Train Loss: 1.453618  Train Epoch: 3 [12800/50000 (26%)] Train Loss: 1.258015  Train Epoch: 10 [38400/50000 (77%)] Train Loss: 0.788162 Train Epoch: 10 [44800/50000 (90%)] Train Loss: 1.083434 [EPOCH: 10], Test Loss: 0.0314, Test Accuracy: 65.40 % |