rESNET

# 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. Data Augmentation이 적용된 CIFAR10 데이터 다운로드 (Train set, Test set 분리하기) '''  train\_dataset = datasets.CIFAR10(root = "../data/CIFAR\_10",                                    train = True,                                    download = True,                                    transform = transforms.Compose([                                      transforms.RandomHorizontalFlip(),                                      transforms.ToTensor(),                                      transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]))  test\_dataset = datasets.CIFAR10(root = "../data/CIFAR\_10",                                  train = False,                                  transform = transforms.Compose([                                      transforms.RandomHorizontalFlip(),                                      transforms.ToTensor(),                                      transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]))  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. ResNet 모델 설계하기 '''  class BasicBlock(nn.Module):      def \_\_init\_\_(self, in\_planes, planes, stride = 1):          super(BasicBlock, self).\_\_init\_\_()          self.conv1 = nn.Conv2d(in\_planes, planes, kernel\_size = 3, stride = stride, padding = 1, bias = False)          self.bn1 = nn.BatchNorm2d(planes)          self.conv2 = nn.Conv2d(planes, planes, kernel\_size = 3, stride = 1, padding = 1, bias = False)          self.bn2 = nn.BatchNorm2d(planes)            self.shortcut = nn.Sequential()          if stride != 1 or in\_planes != planes:              self.shortcut = nn.Sequential(                  nn.Conv2d(in\_planes, planes, kernel\_size = 1, stride = stride, bias = False),                  nn.BatchNorm2d(planes))        def forward(self, x):          out = F.relu(self.bn1(self.conv1(x)))          out = self.bn2(self.conv2(out))          out += self.shortcut(x)          out = F.relu(out)          return out    class ResNet(nn.Module):      def \_\_init\_\_(self, num\_classes = 10):          super(ResNet, self).\_\_init\_\_()          self.in\_planes = 16            self.conv1 = nn.Conv2d(3, 16, kernel\_size = 3, stride = 1, padding = 1, bias = False)          self.bn1 = nn.BatchNorm2d(16)          self.layer1 = self.\_make\_layer(16, 2, stride = 1)          self.layer2 = self.\_make\_layer(32, 2, stride = 2)          self.layer3 = self.\_make\_layer(64, 2, stride = 2)          self.linear = nn.Linear(64, num\_classes)        def \_make\_layer(self, planes, num\_blocks, stride):          strides = [stride] + [1] \* (num\_blocks  - 1)          layers = []          for stride in strides:              layers.append(BasicBlock(self.in\_planes, planes, stride))              self.in\_planes = planes          return nn.Sequential(\*layers)        def forward(self, x):          out = F.relu(self.bn1(self.conv1(x)))          out = self.layer1(out)          out = self.layer2(out)          out = self.layer3(out)          out = F.avg\_pool2d(out, 8)          out = out.view(out.size(0), -1)          out = self.linear(out)          return out  ''' 7. Optimizer, Objective Function 설정하기 '''  model = ResNet().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)  criterion = nn.CrossEntropyLoss()  print(model)  ''' 8. ResNet 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. ResNet 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: 2.239057  Train Epoch: 1 [6400/50000 (13%)] Train Loss: 1.676803  Train Epoch: 1 [12800/50000 (26%)] Train Loss: 1.528677  Train Epoch: 1 [19200/50000 (38%)] Train Loss: 1.192114  Train Epoch: 1 [25600/50000 (51%)] Train Loss: 1.292242  Train Epoch: 1 [32000/50000 (64%)] Train Loss: 0.983950  Train Epoch: 1 [38400/50000 (77%)] Train Loss: 1.140914  Train Epoch: 1 [44800/50000 (90%)] Train Loss: 0.929763  [EPOCH: 1], Test Loss: 0.0338, Test Accuracy: 60.81 %  Train Epoch: 2 [0/50000 (0%)] Train Loss: 1.265161  Train Epoch: 2 [6400/50000 (13%)] Train Loss: 1.060236  Train Epoch: 2 [12800/50000 (26%)] Train Loss: 1.030123  Train Epoch: 2 [19200/50000 (38%)] Train Loss: 1.025596  Train Epoch: 2 [25600/50000 (51%)] Train Loss: 0.866592  Train Epoch: 2 [32000/50000 (64%)] Train Loss: 0.703173  Train Epoch: 2 [38400/50000 (77%)] Train Loss: 0.855952  Train Epoch: 2 [44800/50000 (90%)] Train Loss: 1.112138  [EPOCH: 2], Test Loss: 0.0283, Test Accuracy: 67.69 %  Train Epoch: 3 [0/50000 (0%)] Train Loss: 1.007991  Train Epoch: 3 [6400/50000 (13%)] Train Loss: 0.525277  Train Epoch: 3 [12800/50000 (26%)] Train Loss: 0.661529  Train Epoch: 10 [38400/50000 (77%)] Train Loss: 0.445151  Train Epoch: 10 [44800/50000 (90%)] Train Loss: 0.396798  [EPOCH: 10], Test Loss: 0.0172, Test Accuracy: 81.10 % |