Artificial Intelligence Lab Work (4)

レポート解答用紙 (Report Answer Sheet)

学生証番号 (Student ID): 18521071

名前(Name): **ズオン・ミン・ルオン(Duong Minh Luong)**

問題1.

|  |
| --- |
| (プログラム)  ##ライブラリの読み込み  import numpy as np  import matplotlib.pyplot as plt  import torch  import torch.nn.functional as F  import torchvision as tv  #訓練データとテストデータの読み込み(初めて実行するときはデータをネットからダウンロードする)  train\_dataset=tv.datasets.MNIST(root="./",train=True,transform=tv.transforms.ToTensor(),download=True)  test\_dataset=tv.datasets.MNIST(root="./",train=False,transform=tv.transforms.ToTensor(),download=True)  #訓練データとテストデータのミニバッチ処理・ミニバッチサイズ=100・データの順番をシャッフル  train\_loader=torch.utils.data.DataLoader(dataset=train\_dataset,batch\_size=100,shuffle=True)  test\_loader=torch.utils.data.DataLoader(dataset=test\_dataset,batch\_size=100,shuffle=False)  MODELNAME = 'mnist.model'  EPOCH = 10  DEVICE = 'cuda' if torch.cuda.is\_available() else 'cpu'  #GPUが利用可能ならDEVICE=”cuda”  #CPUを利用するのならDEVICE=”cpu  class MNIST(torch.nn.Module):  def \_\_init\_\_(self):  super(MNIST,self).\_\_init\_\_()  self.l1 = torch.nn.Linear(784,300)  self.l2 = torch.nn.Linear(300,300)  self.l3 = torch.nn.Linear(300,10)  def forward(self,x): #forward計算のときに forward()が呼ばれる。同時にネットワークを構築  h = F.relu(self.l1(x))  h = F.relu(self.l2(h))  y = self.l3(h)  return y  def train\_MNIST():  model = MNIST().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters())  for epoch in range(EPOCH):  loss =0  for images, labels in train\_loader: #データをミニバッチサイズに切り出す  images = images.view(-1,28\*28).to(DEVICE)  #(100✕1✕28✕28)から (100× 784)に変形  labels = labels.to(DEVICE)  optimizer.zero\_grad()  y = model(images)  batchloss =F.cross\_entropy(y, labels)  batchloss.backward()  optimizer.step()  loss = loss + batchloss.item()  print('epoch',epoch,': loss',loss)  torch.save(model.state\_dict(),MODELNAME)  def test\_MNIST():  total = len(test\_loader.dataset)  correct = 0  model = MNIST().to('cpu')  model.load\_state\_dict(torch.load(MODELNAME)) #ファイルに保存したモデルをロード  model.eval()  for images, labels in test\_loader: #テストデータに対してループ  images = images.view(-1,28\*28).to('cpu')  y = model(images)  pred\_labels = y.max(dim=1)[1]  correct = correct + (pred\_labels ==labels).sum()  print('correct:',correct.item())  print('total:',total)  print('accuracy:',(correct.item()/float(total)))  import time  start = time.time()  train\_MNIST()  print("Completed:", time.time()-start)  test\_MNIST() |

|  |
| --- |
| (実行結果) |

問題2

|  |
| --- |
| (プログラム)  ##ライブラリの読み込み  import numpy as np  import matplotlib.pyplot as plt  import torch  import torch.nn.functional as F  import torchvision as tv  import time  train\_dataset = tv.datasets.CIFAR10(root="./",train=True,transform=tv.transforms.ToTensor(),download=True)  test\_dataset = tv.datasets.CIFAR10(root="./", train=False,transform=tv.transforms.ToTensor(),download=True)  train\_loader = torch.utils.data.DataLoader(dataset=train\_dataset, batch\_size=100, shuffle=True)  test\_loader = torch.utils.data.DataLoader(dataset=test\_dataset, batch\_size=100, shuffle=False)  MODELNAME = 'CIFAR10.model'  EPOCH = 10  DEVICE = 'cuda' if torch.cuda.is\_available() else 'cpu'  class CIFAR10(torch.nn.Module):  def \_\_init\_\_(self):  super(CIFAR10,self).\_\_init\_\_()  self.l1 = torch.nn.Linear(3\*32\*32,300)  self.l2 = torch.nn.Linear(300,300)  self.l3 = torch.nn.Linear(300,10)  def forward(self,x):  h = F.relu(self.l1(x))  h = F.relu(self.l2(h))  y = self.l3(h)  return y  def train\_CIFAR10():  model = CIFAR10().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters())  for epoch in range(EPOCH):  loss =0  for images, labels in train\_loader:  images = images.view(-1,3\*32\*32).to(DEVICE)  labels = labels.to(DEVICE)  optimizer.zero\_grad()  y = model(images)  batchloss =F.cross\_entropy(y, labels)  batchloss.backward()  optimizer.step()  loss = loss + batchloss.item()  print('epoch',epoch,': loss',loss)  torch.save(model.state\_dict(),MODELNAME)  def test\_CIFAR10():  total = len(test\_loader.dataset)  correct = 0  model = CIFAR10().to('cpu')  model.load\_state\_dict(torch.load(MODELNAME))  model.eval()  for images, labels in test\_loader:  images = images.view(-1,3\*32\*32).to('cpu')  y = model(images)  pred\_labels = y.max(dim=1)[1]  correct = correct + (pred\_labels ==labels).sum()  print('correct:',correct.item())  print('total:',total)  print('accuracy:',(correct.item()/float(total)))  start = time.time()  train\_CIFAR10()  print("Completed:", time.time()-start)  test\_CIFAR10() |

|  |
| --- |
| (実行結果) |

問題3

|  |
| --- |
| (プログラム)  ##ライブラリの読み込み  import numpy as np  import matplotlib.pyplot as plt  import torch  import torch.nn.functional as F  import torchvision as tv  import time  train\_dataset = tv.datasets.CIFAR10(root="./", train=True,transform=tv.transforms.ToTensor(),download=True)  test\_dataset = tv.datasets.CIFAR10(root="./", train=False,transform=tv.transforms.ToTensor(),download=True)  train\_loader = torch.utils.data.DataLoader(dataset=train\_dataset, batch\_size=100, shuffle=True)  test\_loader = torch.utils.data.DataLoader(dataset=test\_dataset, batch\_size=100, shuffle=False)  MODELNAME = 'CIFAR10\_2D.model'  EPOCH = 10  DEVICE = 'cuda' if torch.cuda.is\_available() else 'cpu'  class CIFAR10\_2D(torch.nn.Module):  def \_\_init\_\_(self):  super(CIFAR10\_2D,self).\_\_init\_\_()  self.l1 = torch.nn.Conv2d(3,16,5)  self.l2 = torch.nn.Linear(16\*28\*28,300)  self.l3 = torch.nn.Linear(300,10)  def forward(self,x):  h = F.relu(self.l1(x))  h = torch.flatten(h,start\_dim=1)  h = F.relu(self.l2(h))  y = self.l3(h)  return y  def train\_2D():  model = CIFAR10\_2D().to(DEVICE)  optimizer = torch.optim.Adam(model.parameters())  for epoch in range(EPOCH):  loss =0  for images, labels in train\_loader:  images = images.view(-1,3,32,32).to(DEVICE)  labels = labels.to(DEVICE)  optimizer.zero\_grad()  y = model(images)  batchloss =F.cross\_entropy(y, labels)  batchloss.backward()  optimizer.step()  loss = loss + batchloss.item()  print('epoch',epoch,': loss',loss)  torch.save(model.state\_dict(),MODELNAME)  def test\_2D():  total = len(test\_loader.dataset)  correct = 0  model = CIFAR10\_2D().to('cpu')  model.load\_state\_dict(torch.load(MODELNAME))  model.eval()  for images, labels in test\_loader:  images = images.view(-1,3,32,32).to('cpu')  y = model(images)  pred\_labels = y.max(dim=1)[1]  correct = correct + (pred\_labels ==labels).sum()  print('correct:',correct.item())  print('total:',total)  print('accuracy:',(correct.item()/float(total)))  start = time.time()  train\_2D()  print("Completed:", time.time()-start)  test\_2D() |

|  |
| --- |
| (実行結果) |