Assignment 11

# Part 1 – Saving the Model

## Code

# DL20A.py CS5173/6073 cheng 2023  
# 2D LeNet5 on MNIST  
# following d2l 7.6.1  
# Usage: python DL20A.py  
  
import torch  
import torchvision  
import torch.nn as nn  
import numpy as np  
import random  
import time  
  
mnist = torchvision.datasets.MNIST('/data/')  
num\_samples = len(mnist)  
x = []  
targets = []  
for i in range(num\_samples):  
 x.append(list(mnist[i][0].getdata()))  
 targets.append(mnist[i][1])  
  
X = torch.tensor(x, dtype=torch.float32)  
X2 = torch.reshape(X, (len(X), 1, 28, 28))  
y = torch.tensor(targets, dtype=torch.long)  
  
model = nn.Sequential(  
 nn.LazyConv2d(6, kernel\_size=5, padding=2), nn.ReLU(),  
 nn.AvgPool2d(kernel\_size=2),  
 nn.LazyConv2d(16, kernel\_size=5), nn.ReLU(),  
 nn.AvgPool2d(kernel\_size=2),  
 nn.Flatten(),  
 nn.LazyLinear(120), nn.ReLU(),  
 nn.LazyLinear(84), nn.ReLU(),  
 nn.LazyLinear(10))  
  
loss\_fun = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters())  
  
batch\_size = 512  
rounds = 1000  
indices = list(range(num\_samples))  
t1 = time.process\_time()  
for i in range(rounds):  
 random.shuffle(indices)  
 batch\_indices = torch.tensor(indices[:batch\_size])  
 Xbatch = X2[batch\_indices]  
 ybatch = y[batch\_indices]  
 o = model(Xbatch)  
 loss = loss\_fun(o, ybatch)  
 if i % 100 == 0:  
 print(i, loss.item())  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
  
t2 = time.process\_time()  
print('Training time', t2 - t1)  
  
# Saving the trained model  
torch.save(model, 'LeNet5\_modern\_ReLU.pt')

# Part 2 – Printing the names and shapes of all trainable parameters in the module.

## Output

Text

Description automatically generated

## Code

# DL23C.py CS5173/6073 cheng 2023  
# ConvMixer on MNIST  
# read the saved DL23B100.zip  
# Usage: python DL23C.py  
  
import torch  
  
saved = torch.load('LeNet5\_modern\_ReLU.pt')  
  
for name, param in saved.named\_parameters():  
 if param.requires\_grad:  
 print(f'Parameter Name: {name}, Shape: {list(param.shape)}')

# Part 3 – Displaying weights of the two LazyConv2d.

## Output

### 0.weight (1st LazyConv2d layer)

A picture containing sky, bathroom, tiled, black

Description automatically generated

### 3.weight (2nd LazyConv2d layer)

A picture containing tiled

Description automatically generated

## Code

# DL23A.py CS5173/6073 cheng 2023  
# ConvMixer parameters  
# Usage: python DL23A.py  
  
import torch  
import matplotlib.pyplot as plt  
  
saved = torch.load('LeNet5\_modern\_ReLU.pt')  
  
patch1 = saved.get\_parameter('0.weight').detach().numpy()  
  
for i in range(6):  
 plt.subplot(2, 3, i + 1)  
 plt.imshow(patch1[i][0], cmap='Blues')  
 plt.xticks([])  
 plt.yticks([])  
plt.show()  
  
patch2 = saved.get\_parameter('3.weight').detach().numpy()  
  
for i in range(6):  
 plt.subplot(4, 4, i + 1)  
 plt.imshow(patch2[i][0], cmap='Blues')  
 plt.xticks([])  
 plt.yticks([])  
plt.show()