BME 646/ ECE695DL: Homework 4

Bianjiang Yang

21 Feb 2022

1 Introduction

The main goal of this homework4:

To start using convolutional layers in a network meant for classifying images. For this homework, we will use the COCO dataset.

To write a image downloader script for the COCO dataset.

To write a dataloader function for the COCO images we will be downloading with our own image downloader script.

2 Methodology

Packages: torch, torch.nn, torch.nn.functional, torchvision.transforms, matplotlib.pyplot, copy, sklearn.metrics, os, sys, glob, seaborn, argparse, requests, PIL, pycocotools.coco, tqdm

Language: Python3

System: Ubuntu 18.04.6 LTS

Instructions for running the code: The code I submitted has been modified to be runable, so you can simply run it by excuting the following script:

(1)hw04_training.py: CUDA_VISIBLE_DEVICES=1 python -u hw04_training.py

(2)hw04_validation.py: CUDA_VISIBLE_DEVICES=1 python -u hw04_validation.py

(3)hw04_coco_downloader.py:

```
python hw04_coco_downloader.py –root_path /home/bjyang/695/hw4/download/Train/
–coco_json_path /home/bjyang/695/hw4/cocoapi/annotations/instances_train2017.json
–class_list ”refrigerator” ”airplane” ”giraffe” ”cat” ”elephant” ”dog” ”train”
”horse” ”boat” ”truck” –images_per_class 2000
python hw04_coco_downloader.py –root_path /home/bjyang/695/hw4/download/Train/
–coco_json_path /home/bjyang/695/hw4/cocoapi/annotations/instances_train2014.json
–class_list ”refrigerator” ”airplane” ”giraffe” ”cat” ”elephant” ”dog” ”train”
”horse” ”boat” ”truck” –images_per_class 1500
python hw04_coco_downloader.py –root_path /home/bjyang/695/hw4/download/Train/
–coco_json_path /home/bjyang/695/hw4/cocoapi/annotations/instances_val2014.json
–class_list ”refrigerator” ”airplane” ”giraffe” ”cat” ”elephant” ”dog” ”train”
”horse” ”boat” ”truck” –images_per_class 500
```

You should modify the savefig directory or just use the plt.show() function to show the plots.

3 Implementation and Results

network.py

```
import torch.nn as nn
import torch.nn.functional as F

class Net1(nn.Module):
    def __init__(self):
        super(Net1, self).__init__()
        self.conv1 = nn.Conv2d(3, 128, 3)
        # self.conv2 = nn.Conv2d(128, 128, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128*31*31, 1000)
        self.fc2 = nn.Linear(1000, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        # x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 128*31*31)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x


class Net2(nn.Module):
    def __init__(self):
        super(Net2, self).__init__()
        self.conv1 = nn.Conv2d(3, 128, 3)
        self.conv2 = nn.Conv2d(128, 128, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128*14*14, 1000)
        self.fc2 = nn.Linear(1000, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 128*14*14)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

class Net3(nn.Module):
    def __init__(self):
        super(Net3, self).__init__()
        self.conv1 = nn.Conv2d(3, 128, 3, 1, 1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128*32*32, 1000)
        self.fc2 = nn.Linear(1000, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        # x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 128*32*32)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

dataloader.py

```
import torch
import glob
import os
from PIL import Image
import numpy as np
class Hw04_Coco_Dataset(torch.utils.data.Dataset):
  'Characterizes a dataset for PyTorch'
  def __init__(self, datapath, transform):
        'Initialization'
        self.transform = transform
        path_list = sorted(glob.glob(datapath + '/*'))
        self.list_IDs = []
        self.labels = []
        n_class = len(path_list)
        for p in range(len(path_list)):
            # class_list.append(os.path.splitext
            # (os.path.basename(p))[0])
            img_path_list = sorted(glob.glob
                                   (path_list[p] + '/*'))
            self.list_IDs = self.list_IDs + img_path_list
            # base = np.zeros(n_class)
            # base[p] = 1
            self.labels = self.labels + [p] * \
                          len(img_path_list)

  def __len__(self):
        'Denotes the total number of samples'
        return len(self.list_IDs)
  def __getitem__(self, index):
        'Generates one sample of data'
        # Select sample
        img_path = self.list_IDs[index]

        # Load data and get label
        X = self.transform(Image.open(img_path)).\
            to(dtype = torch.float32)
        y = self.labels[index]

        return X, y

# cocodata = Hw04_Coco_Dataset("/home/bjyang/
# 695/hw4/hw04_coco_data/Train")
# print(cocodata.__len__())
# img, label = cocodata.__getitem__(2500)
# print(img)
# print(label)
```

hw04_coco_downloader.py

```
#Running Instruction:
# python hw04_coco_downloader.py --root_path
# /home/bjyang/695/hw4/download/Val/
# --coco_json_path /home/bjyang/695/hw4/cocoapi
# /annotations/instances_val2014.json
# --class_list "refrigerator" "airplane"
# "giraffe" "cat" "elephant" "dog" "train"
# "horse" "boat" "truck" --images_per_class 500

# python hw04_coco_downloader.py --root_path
# /home/bjyang/695/hw4/download/Train/
# --coco_json_path /home/bjyang/695/hw4/cocoapi
# /annotations/instances_train2017.json
# --class_list "refrigerator" "airplane"
# "giraffe" "cat" "elephant" "dog" "train"
# "horse" "boat" "truck" --images_per_class 2000

# python hw04_coco_downloader.py --root_path
# /home/bjyang/695/hw4/download/Train/
# --coco_json_path /home/bjyang/695/hw4/cocoapi/
# annotations/instances_train2014.json
# --class_list "refrigerator" "airplane" "
# giraffe" "cat" "elephant" "dog" "train"
# "horse" "boat" "truck" --images_per_class 1500


import argparse
import json
import ast
import requests
import os
from PIL import Image
from requests.exceptions import \
    ConnectionError, ReadTimeout, \
    TooManyRedirects, MissingSchema, InvalidURL
import logging
from pycocotools.coco import COCO
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm


def Coco_Downloader(args):
    coco = COCO(args.coco_json_path)
    urls = dict.fromkeys(args.class_list)
    folder_dict = dict.fromkeys(args.class_list)

    for cla in args.class_list:
        if not os.path.exists(args.root_path + cla):
            os.makedirs(args.root_path + cla)
        folder_dict[cla] = args.root_path + cla
        catIds = coco.getCatIds(cla)
        imgIds = coco.getImgIds(catIds=catIds)
        imgs = coco.loadImgs(imgIds)
        urls[cla] = [i[’coco_url’] for i in imgs]

    for cla in args.class_list:
        folder = folder_dict[cla]
        url = urls[cla]
        print("Downloading Class " + cla)
        for i in tqdm(range(args.images_per_class)):
            per_url = url[i]
            img_name = per_url.split(’/’)[-1]
            file_path = os.path.join(folder, img_name)

            if os.path.exists(file_path):
                # print("File already exists: " + per_url + "\n " +
                #       "Will skip it and continue to the next one.")
                continue

            try: response = requests.get(per_url, timeout=1)
            except Exception:
                try: response = requests.get(per_url, timeout=1) # try again
                except Exception:
                    print("Tried twice and still no "
                          "response for: " + per_url + "\n " +
                           "Will skip it and continue"
                           " to the next one.")
                    continue


            with open(file_path, 'wb') as im:
                im.write(response.content)

            img = Image.open(file_path)
            img_resize = img.resize((64, 64), Image.BOX)
            img_resize.save(file_path)
        print("Class "+ cla +" Finished!")



#provided
parser = argparse.ArgumentParser(description =
                                 'HW04 COCO downloader')
parser.add_argument('--root_path', required = True,
                    type = str)
parser.add_argument('--coco_json_path', required = True,
                    type = str)
parser.add_argument('--class_list', required = True,
                    nargs='*', type=str,)
parser.add_argument('--images_per_class', required=True,
                    type=int)
args, args_other = parser.parse_known_args()

Coco_Downloader(args)
```

hw04_training.py

```
import torch
import torch.nn as nn
import network
import torchvision.transforms as tvt
import matplotlib.pyplot as plt
import dataloader
import copy

def train(transform, device, lr = 1e-3, momentum = 0.9, epochs = 10,
          batch_size = 10,
          data_path = "/home/bjyang/695/hw4/hw04_coco_data/Train",
          save_path = "/home/bjyang/695/hw4/"):

    coco_data = dataloader.Hw04_Coco_Dataset(data_path,
                                             transform)
    train_data = torch.utils.data.DataLoader(coco_data,
                 batch_size=batch_size, shuffle=True, num_workers=2)
    net1 = network.Net1() # {Net1, Net2, Net3}
    net1 = copy.deepcopy(net1)
    net1 = net1.to(device)
    net2 = network.Net2()  # {Net1, Net2, Net3}
    net2 = copy.deepcopy(net2)
    net2 = net2.to(device)
    net3 = network.Net3()  # {Net1, Net2, Net3}
    net3 = copy.deepcopy(net3)
    net3 = net3.to(device)
    running_loss1 = []
    running_loss2 = []
    running_loss3 = []

    criterion = nn.CrossEntropyLoss()

    optimizer = torch.optim.SGD(net1.parameters(), lr=lr,
                                momentum=momentum)
    print("\n\nStarting training loop1")
    for epoch in range(epochs):
        print("")
        running_loss = 0.0
        for i, data in enumerate(train_data):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            outputs = net1(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
            if (i+1) % 500 == 0:
                print("\n[epoch:%d, batch:%5d] loss: %.3f" %
                      (epoch + 1, i + 1, running_loss / float(500)))
                running_loss1.append(running_loss/float(500))
                running_loss = 0.0
    torch.save(net1.state_dict(), save_path+'net1.pth')

    optimizer = torch.optim.SGD(net2.parameters(), lr=lr,
                                momentum=momentum)
    print("\n\nStarting training loop2")
    for epoch in range(epochs):
        print("")
        running_loss = 0.0
        for i, data in enumerate(train_data):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            outputs = net2(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
            if (i + 1) % 500 == 0:
                print("\n[epoch:%d, batch:%5d] loss: %.3f" %
                      (epoch + 1, i + 1, running_loss / float(500)))
                running_loss2.append(running_loss/float(500))
                running_loss = 0.0
    torch.save(net2.state_dict(), save_path+'net2.pth')

    optimizer = torch.optim.SGD(net3.parameters(),
                                lr=lr, momentum=momentum)
    print("\n\nStarting training loop3")
    for epoch in range(epochs):
        print("")
        running_loss = 0.0
        for i, data in enumerate(train_data):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            outputs = net3(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
```

```
            running_loss += loss.item()
            if (i + 1) % 500 == 0:
                print("\n[epoch:%d, batch:%5d] loss: %.3f" %
                      (epoch + 1, i + 1, running_loss / float(500)))
                running_loss3.append(running_loss/float(500))
                running_loss = 0.0
    torch.save(net3.state_dict(), save_path+'net3.pth')
    return running_loss1, running_loss2, running_loss3


if __name__ == '__main__':
    device  = torch.device('cuda:0')
    transform = tvt.Compose([tvt.ToTensor(),
                             tvt.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
    trainset_path = "/home/bjyang/695/hw4/hw04_coco_data/Train"
    save_path = "/home/bjyang/695/hw4/"
    running_loss1, running_loss2, running_loss3 = \
        train(transform = transform, device = device,
              lr = 1e-3, momentum = 0.9,
          epochs = 10, batch_size = 10,
              data_path = trainset_path, save_path = save_path)

    plt.figure()
    plt.title('Train Loss Comparison')
    plt.xlabel('Per 500 Iterations')
    plt.ylabel('Loss')
    plt.plot(running_loss1, label = 'Net1')
    plt.plot(running_loss2, label = 'Net2')
    plt.plot(running_loss3, label = 'Net3')
    plt.legend(loc='upper right')
    plt.show()
    plt.savefig(save_path + "train_loss" + ".jpg")
```

hw04_validation.py

```
import torch
import torch.nn as nn
import network
import torchvision.transforms as tvt
import matplotlib.pyplot as plt
import dataloader
import copy
import sklearn.metrics
import os
import sys
import glob
import seaborn
# import sklearn.metrics.confusion_matrix
# as confusion_matrix

def test(name, net1, transform, device, lr = 1e-3,
         momentum = 0.9, epochs = 10,
          batch_size = 10, data_path = "/home/bjyang"
          "/695/hw4/hw04_coco_data/Val",
          save_path = "/home/bjyang/695/hw4/"):

    coco_data = dataloader.Hw04_Coco_Dataset\
        (data_path, transform)
    test_data = torch.utils.data.DataLoader\
        (coco_data, batch_size=batch_size,
         shuffle=True, num_workers=2)
    # net1 = network.Net1()
    net1 = copy.deepcopy(net1)
    net1.load_state_dict(torch.load(save_path
                                    + ’net’+name+’.pth’))
    net1 = net1.to(device)
    net1.eval()

    print("\n\nStarting testing loop")
    output_total = []
    label_total = []
    for i, data in enumerate(test_data):
        inputs, labels = data
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = net1(inputs)
        prediction = [torch.argmax(output).cpu()
                      for output in outputs]
        output_total = output_total + prediction
        labels = [label.cpu() for label in labels]
        # print(prediction)
        # print(labels)
        label_total = label_total + labels
    confus_matrix = sklearn.metrics.confusion_matrix\
        (label_total, output_total, labels=[0, 1, 2,
                                            3, 4, 5,
                                            6, 7, 8,
                                            9])
    print(confus_matrix)
    acc = 0
    for i in range(confus_matrix.shape[0]):
        acc += confus_matrix[i][i]
    Accuracy = acc / confus_matrix.sum() * 100
    plt_labels = []
    path_list = sorted(glob.glob(data_path + '/*'))
    for p in path_list:
        plt_labels.append(os.path.splitext(os.path.
                                           basename
                                           (p))[0])
    plt.figure(figsize = (10,7))
    seaborn.heatmap(confus_matrix, annot=True,
                    fmt= 'd', linewidths = .5,
                    xticklabels= plt_labels,
                    yticklabels= plt_labels)
    plt.title("Net" + name + " " + 'Accuracy:'
              +str(Accuracy)+'%')

    # cmd = sklearn.metrics.ConfusionMa
    # trixDisplay(confus_matrix, display_labels=plt_labels)
    # cmd.plot(xticks_rotation=15.0)
    plt.savefig(save_path + "net_"+name+
                "confusion_matrix.jpg")



if __name__ == '__main__':
    device  = torch.device('cuda:0')
    transform = tvt.Compose([tvt.ToTensor(),
                             tvt.Normalize
                             ((0.5, 0.5, 0.5),
                              (0.5, 0.5, 0.5))])
    dataset_path = "/home/bjyang/695/hw4/" \
                   "hw04_coco_data/Val"
    save_path = "/home/bjyang/695/hw4/"

    test(name = '1',  net1= network.Net1(),
         transform = transform, device =
         device, lr = 1e-3, momentum = 0.9,
          epochs = 10, batch_size = 10,
         data_path = dataset_path,
         save_path = save_path)
    test(name = '2',  net1=network.Net2(),
         transform=transform, device=
         device, lr=1e-3, momentum=0.9,
         epochs=10, batch_size=10,
         data_path=dataset_path,
         save_path=save_path)
    test(name = '3',  net1=network.Net3(),
         transform=transform, device=
         device, lr=1e-3, momentum=0.9,
         epochs=10, batch_size=10,
         data_path=dataset_path,
         save_path=save_path)
```

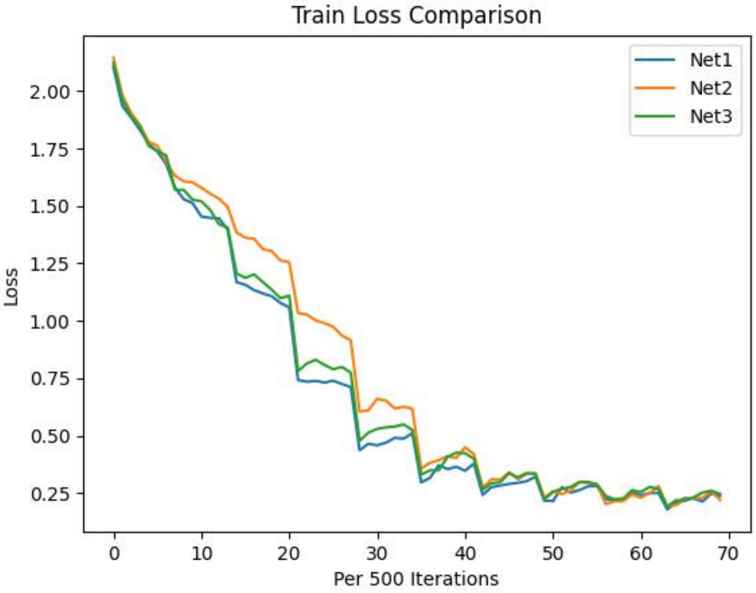
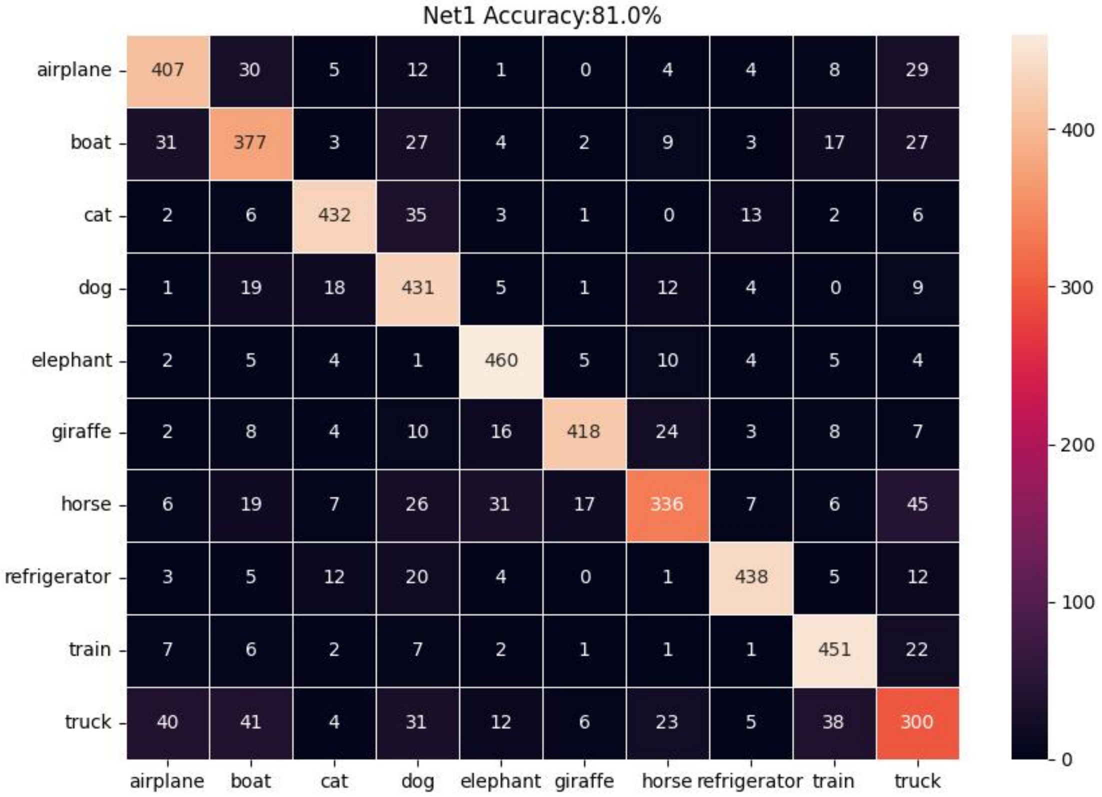
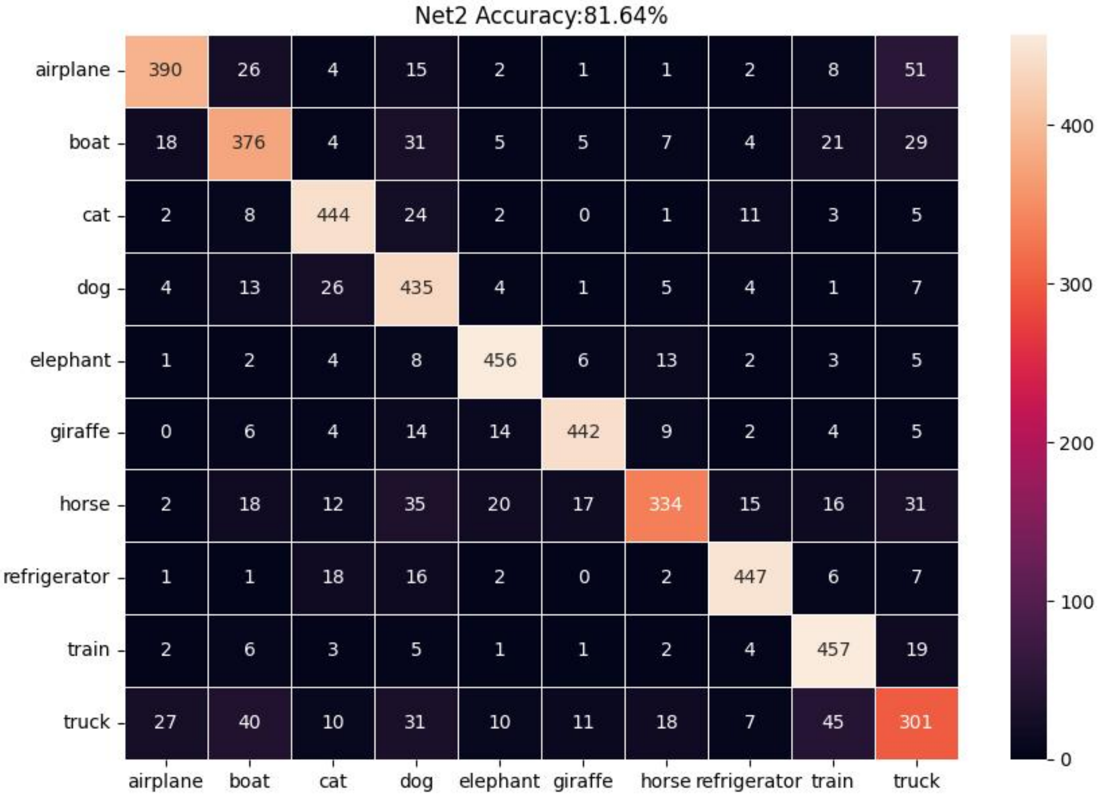


Figure 1: train loss



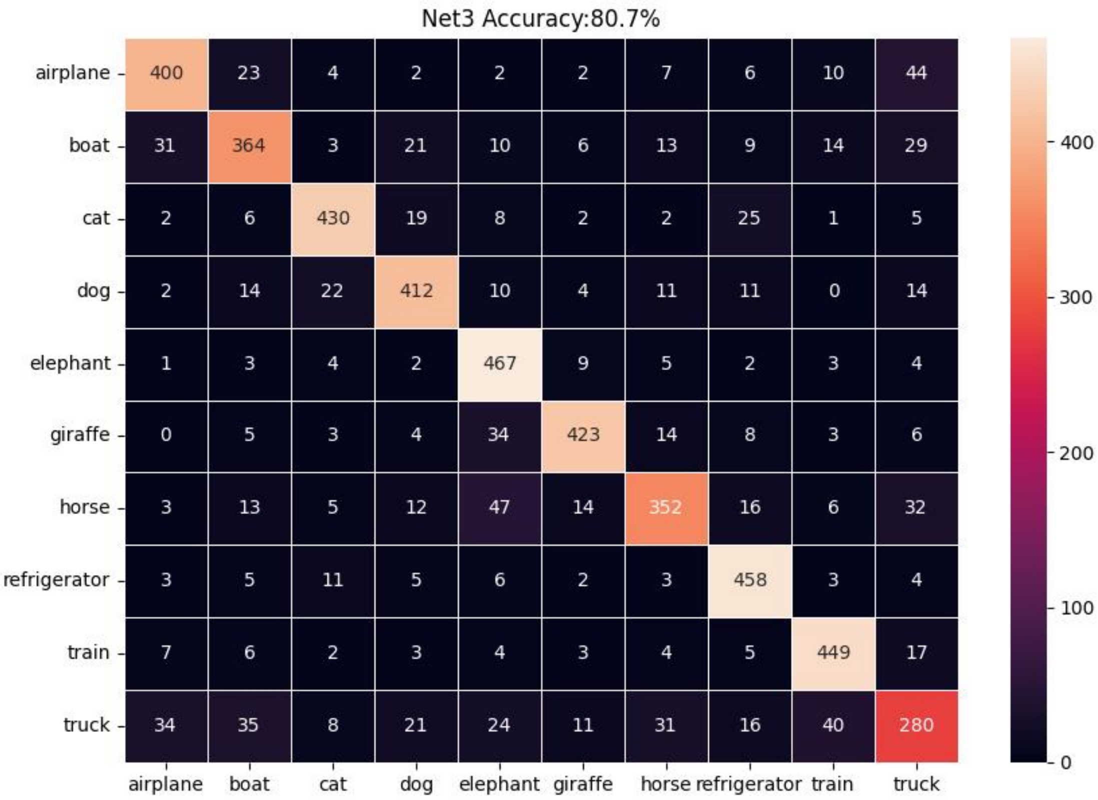
	airplane	boat	cat	dog	elephant	giraffe	horse	refrigerator	train	truck
airplane	407	30	5	12	1	0	4	4	8	29
boat	31	377	3	27	4	2	9	3	17	27
cat	2	6	432	35	3	1	0	13	2	6
dog	1	19	18	431	5	1	12	4	0	9
elephant	2	5	4	1	460	5	10	4	5	4
giraffe	2	8	4	10	16	418	24	3	8	7
horse	6	19	7	26	31	17	336	7	6	45
refrigerator	3	5	12	20	4	0	1	438	5	12
train	7	6	2	7	2	1	1	1	451	22
truck	40	41	4	31	12	6	23	5	38	300

Figure 2: net1 confusion matrix



	airplane	boat	cat	dog	elephant	giraffe	horse	refrigerator	train	truck
airplane	390	26	4	15	2	1	1	2	8	51
boat	18	376	4	31	5	5	7	4	21	29
cat	2	8	444	24	2	0	1	11	3	5
dog	4	13	26	435	4	1	5	4	1	7
elephant	1	2	4	8	456	6	13	2	3	5
giraffe	0	6	4	14	14	442	9	2	4	5
horse	2	18	12	35	20	17	334	15	16	31
refrigerator	1	1	18	16	2	0	2	447	6	7
train	2	6	3	5	1	1	2	4	457	19
truck	27	40	10	31	10	11	18	7	45	301

Figure 3: net2 confusion matrix



	airplane	boat	cat	dog	elephant	giraffe	horse	refrigerator	train	truck
airplane	400	23	4	2	2	2	7	6	10	44
boat	31	364	3	21	10	6	13	9	14	29
cat	2	6	430	19	8	2	2	25	1	5
dog	2	14	22	412	10	4	11	11	0	14
elephant	1	3	4	2	467	9	5	2	3	4
giraffe	0	5	3	4	34	423	14	8	3	6
horse	3	13	5	12	47	14	352	16	6	32
refrigerator	3	5	11	5	6	2	3	458	3	4
train	7	6	2	3	4	3	4	5	449	17
truck	34	35	8	21	24	11	31	16	40	280

Figure 4: net3 confusion matrix

Train Loss

The results in Fig. 1 above shows that:
(1) Net1 loss typcially is less than Net3 and Net2
(2) After 50 × 500 iterations, the network converged.

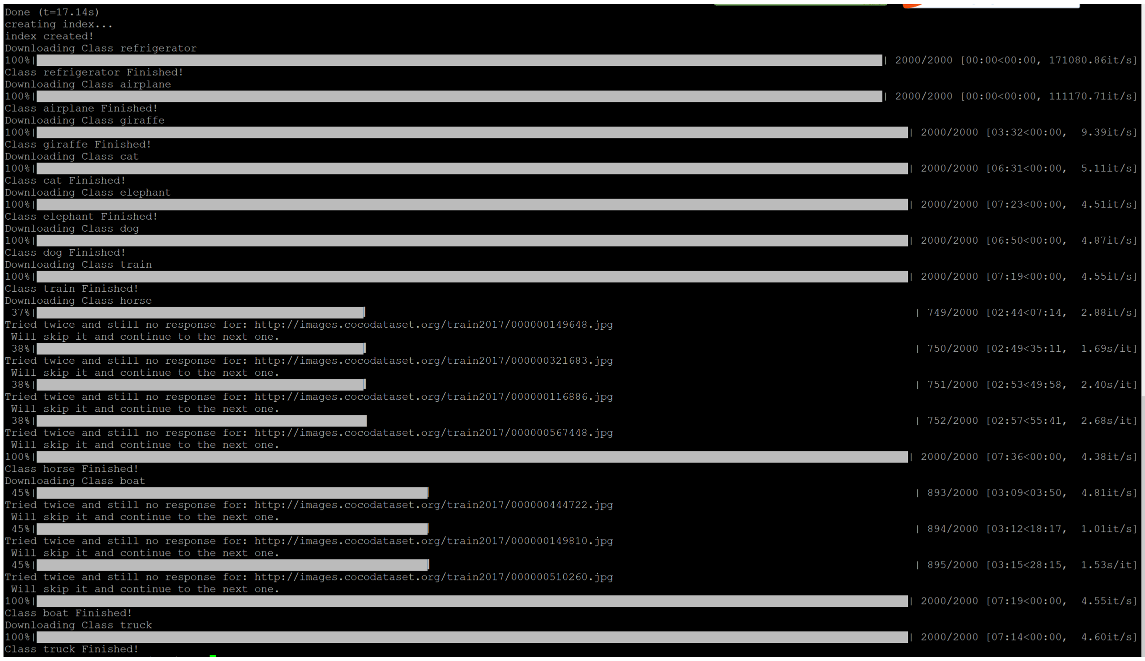
Confusion Matrix

The results in Fig. 2, Fig. 3 and Fig. 4 above shows that:
(1) Net2 has the best performance. So it indicates Net1 may overfit.
(2) Net1, Net2 and Net3 all have an accuracy more than 80%, which indicates most of the prediction is correct.
(3) 'truck' class is typically hard to classify for the networks. 'truck' sometimes is classified to 'train', which may be reasonable because truck and train share some common features so they are hard to be distinguished to some extent. 'elepant' is typically the easiest one to classify.

4 Lessons Learned

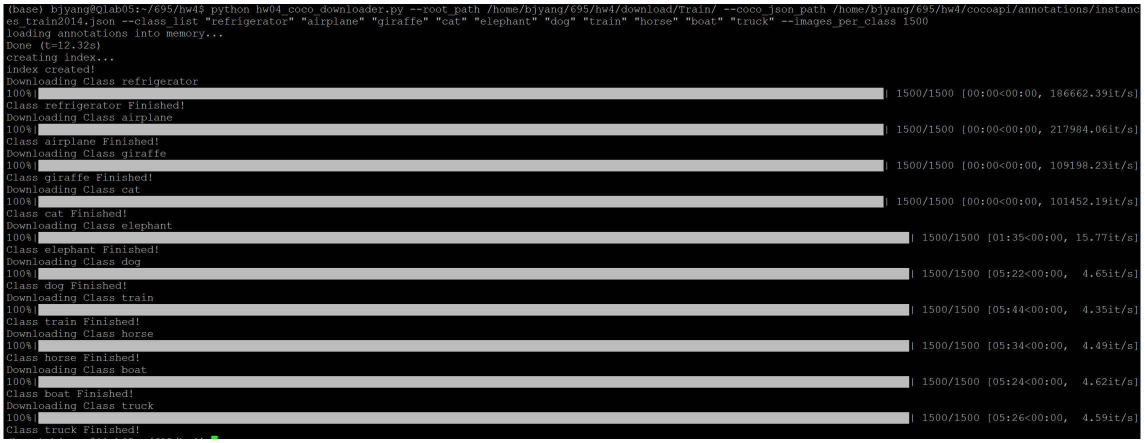
The hurdles faced and the techniques employed to overcome them:

(1) For datadownloader,
We need to deal with some exceptions such as there is no response from the server of COCO. Otherwise, the downloader would not continue downloading other imgs.
We can add tqdm to use the ProgressBar for better visualization of the downloading procedures. There are 3 samples for the downloading process:



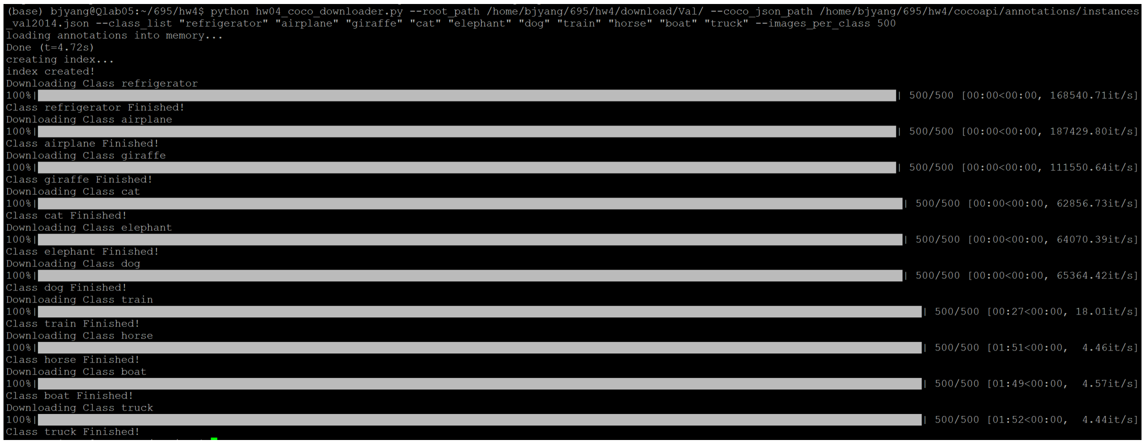
```
Done (t=17.14s)
creating index...
index created!
Downloading Class refrigerator
100%|████████████████████████████████████████| 2000/2000 [00:00<00:00, 171080.86it/s]
Class refrigerator Finished!
Downloading Class airplane
100%|████████████████████████████████████████| 2000/2000 [00:00<00:00, 111170.71it/s]
Class airplane Finished!
Downloading Class giraffe
100%|█████████████████████████████████████████| 2000/2000 [03:32<00:00,  9.39it/s]
Class giraffe Finished!
Downloading Class cat
100%|█████████████████████████████████████████| 2000/2000 [06:31<00:00,  5.11it/s]
Class cat Finished!
Downloading Class elephant
100%|█████████████████████████████████████████| 2000/2000 [07:23<00:00,  4.51it/s]
Class elephant Finished!
Downloading Class dog
100%|█████████████████████████████████████████| 2000/2000 [06:50<00:00,  4.87it/s]
Class dog Finished!
Downloading Class train
100%|█████████████████████████████████████████| 2000/2000 [07:19<00:00,  4.55it/s]
Class train Finished!
Downloading Class horse
 37%|███████████████                           | 749/2000 [02:44<07:14,  2.88it/s]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000149648.jpg
 Will skip it and continue to the next one.
 38%|███████████████                           | 750/2000 [02:49<35:11,  1.69s/it]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000321683.jpg
 Will skip it and continue to the next one.
 38%|███████████████                           | 751/2000 [02:53<49:58,  2.40s/it]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000116886.jpg
 Will skip it and continue to the next one.
 38%|███████████████                           | 752/2000 [02:57<55:41,  2.68s/it]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000567448.jpg
 Will skip it and continue to the next one.
100%|█████████████████████████████████████████| 2000/2000 [07:36<00:00,  4.38it/s]
Class horse Finished!
Downloading Class boat
 45%|██████████████████                        | 893/2000 [03:09<03:50,  4.81it/s]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000444722.jpg
 Will skip it and continue to the next one.
 45%|██████████████████                        | 894/2000 [03:12<18:17,  1.01it/s]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000149810.jpg
 Will skip it and continue to the next one.
 45%|██████████████████                        | 895/2000 [03:15<28:15,  1.53s/it]
Tried twice and still no response for: http://images.cocodataset.org/train2017/000000510260.jpg
 Will skip it and continue to the next one.
100%|█████████████████████████████████████████| 2000/2000 [07:19<00:00,  4.55it/s]
Class boat Finished!
Downloading Class truck
100%|█████████████████████████████████████████| 2000/2000 [07:14<00:00,  4.60it/s]
Class truck Finished!
```

Figure 5: train 2017



```
(base) bjyang@Qlab05:~/695/hw4$ python hw04_coco_downloader.py --root_path /home/bjyang/695/hw4/download/Train/ --coco_json_path /home/bjyang/695/hw4/cocoapi/annotations/instanc
es_train2014.json --class_list "refrigerator" "airplane" "giraffe" "cat" "elephant" "dog" "train" "horse" "boat" "truck" --images_per_class 1500
loading annotations into memory...
Done (t=12.32s)
creating index...
index created!
Downloading Class refrigerator
100%|████████████████████████████████████████████████████████████| 1500/1500 [00:00<00:00, 186662.39it/s]
Class refrigerator Finished!
Downloading Class airplane
100%|████████████████████████████████████████████████████████████| 1500/1500 [00:00<00:00, 217984.06it/s]
Class airplane Finished!
Downloading Class giraffe
100%|████████████████████████████████████████████████████████████| 1500/1500 [00:00<00:00, 109198.23it/s]
Class giraffe Finished!
Downloading Class cat
100%|████████████████████████████████████████████████████████████| 1500/1500 [00:00<00:00, 101452.19it/s]
Class cat Finished!
Downloading Class elephant
100%|██████████████████████████████████████████████████████████████| 1500/1500 [01:35<00:00, 15.77it/s]
Class elephant Finished!
Downloading Class dog
100%|██████████████████████████████████████████████████████████████| 1500/1500 [05:22<00:00,  4.65it/s]
Class dog Finished!
Downloading Class train
100%|██████████████████████████████████████████████████████████████| 1500/1500 [05:44<00:00,  4.35it/s]
Class train Finished!
Downloading Class horse
100%|██████████████████████████████████████████████████████████████| 1500/1500 [05:34<00:00,  4.49it/s]
Class horse Finished!
Downloading Class boat
100%|██████████████████████████████████████████████████████████████| 1500/1500 [05:24<00:00,  4.62it/s]
Class boat Finished!
Downloading Class truck
100%|██████████████████████████████████████████████████████████████| 1500/1500 [05:26<00:00,  4.59it/s]
Class truck Finished!
```

Figure 6: train 2014



```
(base) bjyang@Qlab05:~/695/hw4$ python hw04_coco_downloader.py --root_path /home/bjyang/695/hw4/download/Val/ --coco_json_path /home/bjyang/695/hw4/cocoapi/annotations/instances
_val2014.json --class_list "refrigerator" "airplane" "giraffe" "cat" "elephant" "dog" "train" "horse" "boat" "truck" --images_per_class 500
loading annotations into memory...
Done (t=4.72s)
creating index...
index created!
Downloading Class refrigerator
100%|████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 168540.71it/s]
Class refrigerator Finished!
Downloading Class airplane
100%|████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 187429.80it/s]
Class airplane Finished!
Downloading Class giraffe
100%|████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 111550.64it/s]
Class giraffe Finished!
Downloading Class cat
100%|█████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 62856.73it/s]
Class cat Finished!
Downloading Class elephant
100%|█████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 64070.39it/s]
Class elephant Finished!
Downloading Class dog
100%|█████████████████████████████████████████████████████████████| 500/500 [00:00<00:00, 65364.42it/s]
Class dog Finished!
Downloading Class train
100%|██████████████████████████████████████████████████████████████| 500/500 [00:27<00:00, 18.01it/s]
Class train Finished!
Downloading Class horse
100%|██████████████████████████████████████████████████████████████| 500/500 [01:51<00:00,  4.46it/s]
Class horse Finished!
Downloading Class boat
100%|██████████████████████████████████████████████████████████████| 500/500 [01:49<00:00,  4.57it/s]
Class boat Finished!
Downloading Class truck
100%|██████████████████████████████████████████████████████████████| 500/500 [01:52<00:00,  4.44it/s]
Class truck Finished!
```

Figure 7: val 2014

(2) For dataloader,
We need to output the label with shape (1,) for one sample, which means the labels are int numbers ranging (0,9). However, the outputs of the network are one-hot-vectors with shape(10,). nn.CrossEntropyLoss() can accept such two inputs.

(3) For hw04_training,
We need to normalize the data.

(4) For hw04_validation,
We need to eval the model to keep the model parameters static.
seaborn is a better tool than sklearn to plot the heatmap.
Confusion Matrix should be computed on CPU because of sklearn.

5 Suggested Enhancements

To the best of my understanding, this task emphasize more on downloader because it takes more time to accomplish. Also, most of the time was consumed in the ploting and some unimportant part debugging. So I suggest you could give a colab file and let the students fill necessary blanks.