附件：软件代码

1. 汽车数据集制作程序

DatasetMake.py

```
import os
import time
import random
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

# 数据集图片重命名
# 按照：分类编号_序号.图片格式
def Rename(path, imgclass):
    num = 1
    for filename in os.listdir(path):
        hz = filename.split('.')[-1]
        hz = '.'+hz
        hz = path + r'/' + imgclass + r'_' + str(num) + hz
        os.rename(path+r'/'+filename, hz)
        num+=1
        time.sleep(0.1)
        if num%50==0 :
            print(num)
    print(num)

N = 227
# 制作数据集：存储在 car 文件中
# 每张图片存储三行,每行一个颜色通道,个数为 227*227
# 通道顺序为 RGB
def CarDataMake(path,savepath):
    filecnt = 0
    lstdir = os.listdir(path)
    random.shuffle(lstdir)
    lstlen = len(lstdir)
    n = 1
    while True:
        savep = savepath + str(n) + '.car'
        n+=1
        m = 0
        with open(savep, 'w', encoding='ascii') as fp:
            while m<400 :
```

```
                    if filecnt>=lstlen:
                        return
                    img = Image.open(path + r'/' + lstdir[filecnt])
                    print(img)
                    tmpimg = img.resize((N,N), Image.ANTIALIAS)
                    print(tmpimg)
                    tmpimg = np.transpose(tmpimg, (2,0,1))
                    img = np.array(tmpimg)
                    print(img.shape, filecnt)
                    fp.write(lstdir[filecnt].split('_')[0]+'\n')
                    for i in range(3):
                        tmplist = img[i].reshape(N*N)
                        tmpstr = ','.join([str(k) for k in tmplist])
                        fp.write(tmpstr+'\n')
                    filecnt+=1
                    m+=1
                    time.sleep(0.2)
            print('filecnt: ',filecnt)

# 加载数据集,并显示前 num 个
def CarDataLoad(path, num, plabel=False):
    with open(path, 'r', encoding='ascii') as fp:
        data = []
        k = 0
        n = 0
        for line in fp:
            k += 1
            if k==1:
                c = int(line)
                if plabel==True:
                    print(c)
            else:
                img = np.array(line.split(','), dtype = int)
                data.append(img.reshape(N,N))
                if k==4:
                    n += 1
                    if n==num:
                        print(c)
                        data = np.transpose(data, (1,2,0))
                        plt.imshow(data)
                        plt.show()
                        return
                    data = []
                    k = 0
```

```
if __name__ == "__main__":
    rename     = 0
    datamake = 0
    dataload = 1
    if rename:
        path = r'F:\毕设\程序\dataset_pic\3'
        Rename(path, '3')
    if datamake:
        path = r'F:\毕设\程序\dataset_pic\all'
        save = r'F:\毕设\程序\dataset\train\\'
        CarDataMake(path,save)
    if dataload:
        path = r'F:\毕设\程序\dataset\test\car19.car'
        CarDataLoad(path, 4, plabel=False)
```

2. AlexNet 模型定义与数据加载函数

AlexNet_model.py

```
import torch
import torch.nn as nn
from collections import OrderedDict
from torch.utils.data import Dataset
import numpy as np
import matplotlib.pyplot as plt

class AlexNet(nn.Module):
    """
    原网络为双 GPU,此处全部只实现一半
    Input - 3x227x227
    C1 - 48@55x55 (11x11 kernel)(4 stride)
    ReLU1
    S1 - 48@27x27 (3x3 kernel, stride 2) Subsampling
    LRN
    C2 - 128@27x27 (5x5 kernel)(1 stride)(2 padding)
    ReLU2
    S2 - 128@13x13 (3x3 kernel, stride 2) Subsampling
    LRN
    C3 - 192@13x13 (3x3 kernel)(1 stride)(1 padding)
    ReLU3
    C4 - 192@13x13 (3x3 kernel)(1 stride)(1 padding)
    ReLU4
    C5 - 128@13x13 (3x3 kernel)(1 stride)(1 padding)
    ReLU5
    S5 - 128@6x6 (3x3 kernel, stride 2) Subsampling
    C6 - 2048@1x1 (6x6 kernel)
    ReLU6
```

```
        Dropout (p=0.5)
        F7 - 2048
        ReLU7
        Dropout (p=0.5)
        F8 - 6 (Output)
        """
        def __init__(self):
            super(AlexNet, self).__init__()
            self.convnet = nn.Sequential(OrderedDict([
                ('C1    ', nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2)),
                ('ReLU1', nn.ReLU(inplace=True)),
                ('S1    ', nn.MaxPool2d(kernel_size=3, stride=2)),
                ('C2    ', nn.Conv2d(64, 192, kernel_size=5, padding=2)),
                ('ReLU3', nn.ReLU(inplace=True)),
                ('S2    ', nn.MaxPool2d(kernel_size=3, stride=2)),
                ('C3    ', nn.Conv2d(192, 384, kernel_size=3, padding=1)),
                ('ReLU3', nn.ReLU(inplace=True)),
                ('C4    ', nn.Conv2d(384, 256, kernel_size=3, padding=1)),
                ('ReLU4', nn.ReLU(inplace=True)),
                ('C5    ', nn.Conv2d(256, 256, kernel_size=3, padding=1)),
                ('ReLU5', nn.ReLU(inplace=True)),
                ('S5    ', nn.MaxPool2d(kernel_size=3, stride=2)),
                ('Avg6 ', nn.AdaptiveAvgPool2d((6, 6))),
                ('Drop6', nn.Dropout()),
            ]))
            self.fc = nn.Sequential(OrderedDict([
                ('F7    ', nn.Linear(256*6*6, 1024)),
                ('ReLU7', nn.ReLU(inplace=True)),
                ('Drop7', nn.Dropout()),
                ('F8    ', nn.Linear(1024, 6)),
            ]))

        def forward(self, x):
            x = self.convnet(x)
            x = torch.flatten(x, 1)
            x = self.fc(x)
            return x

    class CarDataset(Dataset):
        """car dataset load"""

        def __init__(self, datapath, train=True):
            """
            Args:
                datapath:filename
                train(optional):traindata or testdata
```

```
        """
        self.label = []
        self.image = []
        self.train = train
        self.len = 0
        self.N = 227

        # 加载数据集
        tmp = []
        k = 0
        if train:
            with open(datapath, 'r', encoding='ascii') as fp:
                for line in fp:
                    k += 1
                    if k==1 :
                        self.label.append(int(line))
                    else:
                        img = np.array(line.split(','), dtype = float)
                        tmp.append(img.reshape(self.N, self.N))
                        if k==4:
                            self.image.append(tmp)
                            tmp = []
                            k = 0
        else:
            with open(datapath, 'r', encoding='ascii') as fp:
                for line in fp:
                    k += 1
                    img = np.array(line.split(','), dtype = float)
                    tmp.append(img.reshape(self.N, self.N))
                    if k==3:
                        self.image.append(tmp)
                        tmp = []
                        k = 0
        self.image = torch.from_numpy(np.array(self.image))
        self.len = len(self.image)

    def __len__(self):
        return self.len

    def __getitem__(self, idx):
        if self.train:
            return (self.image[idx].float(), torch.tensor(self.label[idx]))
        else:
            return self.image[idx]

    def shape(self):
```

```
            print(self.image.shape)

        def show(self, idx):
            if self.train:
                print(self.label[idx])
            img = np.array(self.image[idx], dtype=int)
            img = np.transpose(img, (1,2,0))
            plt.imshow(img)
            plt.show()


if __name__ == '__main__':

        net = AlexNet()
        print(net)
```

3. 模型训练程序

AlexNet_train.py

```
from AlexNet_model import AlexNet
from AlexNet_model import CarDataset

import torch
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# run device
device = torch.device('cpu')
if torch.cuda.is_available():
        print('device: gpu {}'.format(torch.cuda.get_device_name()))
        device = torch.device('cuda')
else:
        print('device: cpu')
        device = torch.device('cpu')
print('device:', device)

# start
# code could be write follows
trainfilename = r'F:\毕设\程序\dataset\train\car'
modelfilename = r'F:\毕设\程序\dataset\alexnet6-08521.pt'
# train
def train(model, epoch):
        BATCHSIZE = 128
        model.train()
        learning_rate = 1e-3
        optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

```
    criterion = torch.nn.CrossEntropyLoss().to(device)
    loss_list = []
    for i in range(epoch):
        for j in range(15):
            # data load
            filename = trainfilename + str(j+1) + '.car'
            print(filename)
            data_train = CarDataset( filename, train=True)
            data_train_loader = DataLoader(data_train, batch_size=BATCHSIZE, shuffle=True)
            for k,(img,lab) in enumerate(data_train_loader):
                img = img.to(device)
                lab = lab.to(device)
                output = model(img)
                loss = criterion(output, lab)
                loss_n = loss.detach().cpu().item()

                print('Train - Epoch %d, Batch: %d, Num: %d, Loss: %f' % (i+1, k,
k*BATCHSIZE, loss_n))
                loss_list.append(loss_n)

                optimizer.zero_grad()
                loss.backward()
                optimizer.step()
    plt.plot(loss_list)
    plt.show()


if __name__ == '__main__':
    alexnet = AlexNet()
    alexnet.load_state_dict(torch.load(modelfilename))
    alexnet.to(device)
    train(alexnet,10)
    print('train finish...')
    # save
    modelfilename = r'F:\毕设\程序\dataset\alexnet8.pt'
    torch.save(alexnet.state_dict(), modelfilename)
    print('save finish...')
```

4. 模型测试程序[检验识别准确率]

AlexNet_test.py

```
from AlexNet_model import AlexNet
from AlexNet_model import CarDataset

import torch
from torch.utils.data import DataLoader
```

```
import matplotlib.pyplot as plt

# run device
device = torch.device('cpu')
if torch.cuda.is_available():
    print('device: gpu {}'.format(torch.cuda.get_device_name()))
    device = torch.device('cuda')
else:
    print('device: cpu')
    device = torch.device('cpu')
print('device:', device)
# start
# code could be write follows
testfilename    = r'F:\毕设\程序\dataset\test\car'
modelfilename = r'F:\毕设\程序\dataset\alexnet6.pt'

# run test
def run(model,printerror=False):
    model.eval()
    criterion = torch.nn.CrossEntropyLoss().to(device)
    lst = [0 for i in range(6)]
    ten = [torch.tensor([i]).to(device) for i in range(6)]
    cnt1,cnt2 = 0,0
    for i in range(4):
        filename = testfilename + str(i+16) + '.car'
        print(filename)
        # data load
        data_test = CarDataset(filename, train=True)
        data_test_loader = DataLoader(data_test)
        for img,lab in data_test_loader:
            img = img.to(device)
            output = model(img)
            for j in range(6):
                lst[j]=criterion(output, ten[j])
            gus = lst.index(min(lst))
            lab = lab.item()
            cnt1+=1
            if gus!=lab:
                cnt2+=1
                if printerror:
                    print('Num:{}\tlab:{}\tgus:{}'.format(cnt1, gus, lab))
    print('Total:{}\tError:{}\tAccuracy:{}'.format(cnt1, cnt2, 1-cnt2/cnt1))


if __name__ == '__main__':
    # load
```

```
    load_model = AlexNet()
    load_model.load_state_dict(torch.load(modelfilename))
    load_model.to(device)
    # run
    run(load_model,printerror=False)
```

5. 车型识别窗口应用程序

car_app.py

```
import torch
import numpy as np
from PIL import Image

import tkinter as tk
import tkinter.filedialog as filedialog
from AlexNet_model import AlexNet

# 待识别图片所在路径
picname = ""
# 模型路径
modelfilename = r'F:\毕设\程序\dataset\alexnet6-08521.pt'
carclass = ['公交车', '货车', '客运车', '面包车', '皮卡车', '小轿车']
# 加载模型
print("加载模型...")
load_model = AlexNet()
load_model.load_state_dict(torch.load(modelfilename))
load_model.eval()
criterion = torch.nn.CrossEntropyLoss()
lst = [0 for i in range(6)]
six = [torch.tensor([i]) for i in range(6)]
print("加载完毕...")

def select_pic():
    global picname
    picname=filedialog.askopenfilename()
    print(picname)

def start():
    img = Image.open(picname)
    tmpimg = img.resize((227,227), Image.ANTIALIAS)
    tmpimg = np.transpose(tmpimg, (2,0,1))
    img = np.array(tmpimg, dtype=float)
    img = torch.from_numpy(np.array([img]))
    img = img.type(torch.FloatTensor)
    output = load_model(img)
    for j in range(6):
```

```
            lst[j]=criterion(output, six[j])
        gus = lst.index(min(lst))
        print('预测值: ', carclass[gus])

def end():
        exit()

if __name__ == "__main__":
        gui = tk.Tk()
        gui.title("car gui")
        gui.geometry("200x100+800+60")

        # 创建按钮
        button1 = tk.Button(gui, text="选择图片", command=select_pic, width=10)
        button1.pack()
        button2 = tk.Button(gui, text="开始识别", command=start, width=10)
        button2.pack()
        button3 = tk.Button(gui, text="退出", command=end, width=6)
        button3.pack()

        gui.mainloop()
```