# 车牌识别_学科实践_项目报告

*组长：杜佳钰 37220222203580*
*组员：鄢志轩 37220222203830 温恺宣 37220222203792*

## 1 项目介绍

我们小组本学期选择 “车牌识别” 作为学科实践这门课程的期末项目，希望能够实现接受图车牌正面图片输入后，识别出对应车牌的文字的功能。
之所以选择 “车牌识别”，有以下两个原因：

### 1.1 车牌字符识别技术是智能交通系统中的关键技术

车牌字符识别技术是智能交通系统中的关键技术，具有广泛的应用前景。它通过自动识别车牌上的字符，可以实现对车辆的有效管理和监控。在交通执法、停车管理、高速公路收费、智能停车场、以及安全监控等多个领域，车牌识别技术都发挥着重要作用。
车牌字符识别技术在智能交通系统中的应用前景十分广阔。它不仅提高了交通管理的效率，还改善了公共服务的质量，促进了交通行业的智能化发展。未来，随着技术的进一步完善和普及，车牌识别技术将在更多领域发挥更大的作用，推动智能交通系统的全面升级。

### 1.2 体现我们对于课程内容的掌握和运用

本学期的学科实践课程中我们学习了大数据的一些初级应用，包括数据收集，数据预处理，模型训练等操作来实现对目标的简单多分类。从课程学习的角度来看，在我们的项目中，对车牌进行识别本质上是训练出能够对各个数字，字母和汉字进行分类的模型，得到分类的结果，即为车牌识别的结果。这一过程是非常符合我们本学期学习的知识。车牌上包含的字符类型共有 66 种，可以较好的展现我们的学习成果。同时我们在对车牌处理前需要对输入的车牌图片数据进行一定的预处理操作，这是我们项目的难点，也是亮点。

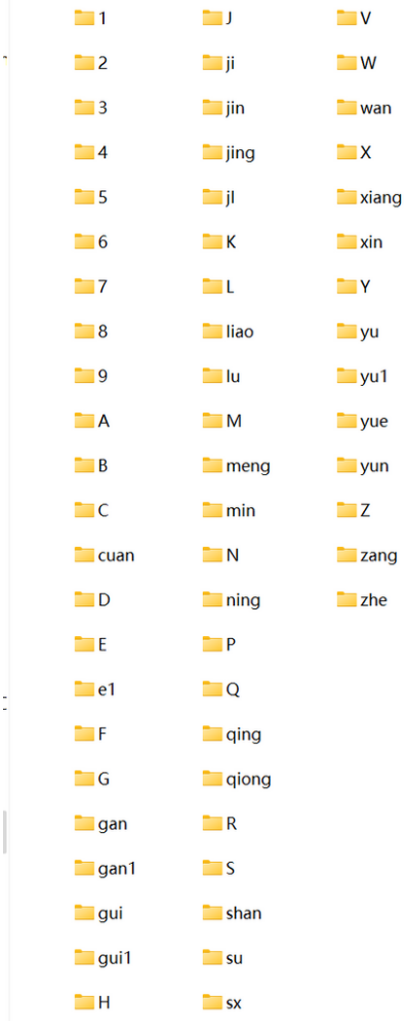
---

## 2 模型准备

### 2.1 数据集选择

在预选题目和老师进行交流沟通后，我们组的选题已经有了较成熟的实现方式，因此我们选择直接使用网络上的公开数据集。通过使用网络公开数据集，我们可以节省收集和整理数据的时间，将更多的时间花在模型的调优处理上，尽可能地提高模型的正确率。
在这个数据集中，字符被分为31个省份，26个英文字母，10个数字，分别进行整理。每个字符集最多达到了1000张图片，最少也有100张左右的图片进行训练。训练的图片数据量较大，便于对每个字符进行较

多轮次的训练，保证了训练质量。




### 2.2 初始模型选择并训练

由于训练的数据量和分类的种类都较多，同时根据之前学习目标分类时的方法，我们选择使用之前作业和项目中处理数据的 dataset 等方法对我们的数据集进行处理。
由于数据集较大，同时分类的种类较多，并且在之前的课程作业中我们比较熟悉 resnet 的模型网络，因此我们选择以paddle平台自带的 resnet101 的训练网络对数据集进行初始的训练，模型及优化器的各个参数如下图所示：

```
model.prepare(paddle.optimizer.Adam(learning_rate=0.0004, parameters=model.parameters()),
              paddle.nn.CrossEntropyLoss(),
              paddle.metric.Accuracy(topk=(1, 2)))  # 返回top1 top2 的准确率



model.fit(train_data=dataset(data_path=target_path, mode='train'),
          eval_data= dataset(data_path=target_path, mode='eval'),
          epochs=10,
          batch_size=128,    #设太大有可能爆内存。
          save_dir="output/",
          save_freq=5,        #保存模型的频率，多少个 epoch 保存一次模型
          log_freq=10,      #日志打印的频率，多少个 step 打印一次日志
          shuffle=True)
```

## 3 模型优化

### 3.1 初始运行结果

在初始训练几次之后，我们发现了训练过程中出现的几个问题：

- 1 正确率较低，初始训练后的 acc 只能达到 0.5-0.6 左右。
- 2 由于数据集较大，因此模型训练的时间较长。即使 epoch 的轮数较小的情况下，也仍然需要花费较长的训练时间。

```
The loss value printed in the log is the current step, and the metric is the avera
Epoch 1/5
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/lay
  warnings.warn(
step 10/11 - loss: 0.5309 - acc_top1: 0.6100 - acc_top2: 0.8400 - 155ms/step
step 11/11 - loss: 0.9312 - acc_top1: 0.6000 - acc_top2: 0.8476 - 154ms/step
save checkpoint at /home/aistudio/output/0
Eval begin...
step 2/2 - loss: 4.7521 - acc_top1: 0.4500 - acc_top2: 0.9000 - 138ms/step
Eval samples: 20
Epoch 2/5
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/lay
  warnings.warn(
step 10/11 - loss: 1.7880 - acc_top1: 0.7200 - acc_top2: 0.9000 - 96ms/step
step 11/11 - loss: 2.8934 - acc_top1: 0.6952 - acc_top2: 0.9048 - 89ms/step
```

```
step 10/16 - loss: 1.6940 - acc_top1: 0.2960 - acc_top2: 0.5760 - 406ms/step
step 16/16 - loss: 1.6923 - acc_top1: 0.2948 - acc_top2: 0.5743 - 397ms/step
Eval begin...
step 3/3 - loss: 0.8043 - acc_top1: 0.3478 - acc_top2: 0.6261 - 259ms/step
Eval samples: 115
Epoch 8/10
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/layer/norm.py:824: UserWarn
  warnings.warn(
step 10/16 - loss: 1.6400 - acc_top1: 0.2380 - acc_top2: 0.5120 - 432ms/step
step 16/16 - loss: 1.5275 - acc_top1: 0.2109 - acc_top2: 0.4803 - 410ms/step
Eval begin...
step 3/3 - loss: 1.7648 - acc_top1: 0.3043 - acc_top2: 0.5304 - 268ms/step
Eval samples: 115
Epoch 9/10
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/layer/norm.py:824: UserWarn
  warnings.warn(
step 10/16 - loss: 1.8647 - acc_top1: 0.2520 - acc_top2: 0.4840 - 413ms/step
step 16/16 - loss: 1.5028 - acc_top1: 0.2783 - acc_top2: 0.5121 - 404ms/step
Eval begin...
step 3/3 - loss: 1.5833 - acc_top1: 0.3652 - acc_top2: 0.6435 - 253ms/step
Eval samples: 115
Epoch 10/10
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/layer/norm.py:824: UserWarn
  warnings.warn(
step 10/16 - loss: 1.5783 - acc_top1: 0.2180 - acc_top2: 0.4640 - 434ms/step
step 16/16 - loss: 1.5446 - acc_top1: 0.2008 - acc_top2: 0.4358 - 419ms/step
Eval begin...
step 3/3 - loss: 1.6850 - acc_top1: 0.2087 - acc_top2: 0.5652 - 249ms/step
Eval samples: 115
save checkpoint at /home/aistudio/output/final
```

### 3.2 更换模型

总结了以上问题，我们猜测可能是由于模型训练层数过多，导致训练时出现了过拟合的问题，导致基于大数量的模型训练后，正确率仍然较低。因此，对问题进行总结后，我们小组认为我们需要更换训练网络，选择一个有合适的层数的模型，以便最快提升正确率与训练效率。
重新考虑数据集的特征后，我们发现：

- 1 训练层数不可过深，避免再次出现模型过拟合的情况
- 2 汉字，英文和数字三类字符之间的字形差异较大，模型训练时提取的特征值差异也不同，比较难将三类之间的字符混淆。因此，即使训练的层数较小，也能够对不同类别之间的数据进行较高正确率的识别。

### 3.3 数据增强

在选择数据集后，我们对数据集内的图片进行了一定的观察，发现训练集中的所有图片都是黑白照片，而非彩色照片。

在查询了相关项目的资料经验，以及结合实际情况进行思考后，我们发现对于字型的分类与图片的颜色本身并没有太多联系。我们可以通过将图片转化为黑白的模式，使得不同字型的差异更加明显，字体图案的边缘更加清晰，同时省略掉了部分实际拍摄中图片可能出现的细节干扰，这样的操作可以对于提高模型的正确率取得一定程度的帮助。
因此，在输入图片的预处理中，我们将图片通过 paddle 的 cv2库 中 cvtColor 的方法将图片转化为黑白图片。并通过像素点的灰度值与 127 进行比较，进行极化操作（小于127的像素点即为白色，0；大于127的像素点即为黑色，256），使得图片界限更加清晰。

```
def __getitem__(self, index):
    """
    获取一组数据
    :param index: 文件索引号
    :return:
    """
    # 第一步打开图像文件并获取label值
    img_path = self.img_paths[index]
    if not os.path.isfile(img_path):
        raise FileNotFoundError(f"File not found or is not a file: {img_path}")

    img = Image.open(img_path)
    if img.mode != 'RGB':
        img = img.convert('RGB')
    img = img.resize((224, 224), Image.BILINEAR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    img = np.array(img).astype('float32')
    img = img.transpose((2, 0, 1)) / 255
    label = self.labels[index]
    label = np.array([label], dtype="int64")
    return img, label
```

### 3.4 数据预处理

我们的项目想要实现的功能是对整张车牌照片进行识别，因此输入的车牌图片是完整的，得到的字符序列也是多个字符串联的结果。
但是训练的模型只能针对每一个字符进行分类，因此在输入需要预测的图片数据后，我们需要对图片进行分割，得到对应不同的字符进行处理。
为了对图片进正确的分割，在查阅了相关项目的资料和经验后，我们采用了像素分割的方法来获取每一块字符的图像：
按列对图片的像素进行统计，遍历二值化图像的每一列，统计每列中白色像素的数量。

- 如果当前列没有白色像素点，且前面没有白色像素点，则这一列不存在字符信息，可以被丢弃；否则代表当前的字符块结束。
- 如果当前列存在白色像素点，且前面没有白色像素点，则新的字符块开始，创建新的 result 数组进行保存；否则直接加入前面创建的字符块

通过这样的方式，可以输入的车牌进行准确度极高的分割，避免了按照比例分割容易发生的错误，正确地对数据进行预处理，便于进行模型的预测。

```python
def process_license_plate(image_path, save_dir):
    # 读取图像
    license_plate = cv2.imread(image_path)
    gray_plate = cv2.cvtColor(license_plate, cv2.COLOR_RGB2GRAY)
    ret, binary_plate = cv2.threshold(gray_plate, 175, 255, cv2.THRESH_BINARY) # ret：阈值，binary_plate：根据阈值处理后的图像数据

    # 按列统计像素分布
    result = []
    for col in range(binary_plate.shape[1]):
        result.append(0)
        for row in range(binary_plate.shape[0]):
            result[col] += binary_plate[row][col] / 255

    # 行上的像素
    width = binary_plate.shape[0]
    ch_width = width / 8

    # 记录车牌中字符的位置
    character_dict = {}
    num = 0
    i = 0
    while i < len(result):
        if result[i] == 0:
            i += 1
        else:
            index = i + 1
            while index < len(result) and result[index] != 0:
                index += 1
            character_dict[num] = [i, index - 1]
            num += 1
            i = index

    # 将每个字符填充，并存储
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    for i in range(8):
        if i == 2:
            continue
        padding = (170 - (character_dict[i][1] - character_dict[i][0])) / 2
        ndarray = np.pad(binary_plate[:, character_dict[i][0]:character_dict[i][1]], ((0, 0), (int(padding), int(padding))), 'constant', constant_values=(0, 0))
        ndarray = cv2.resize(ndarray, (20, 20))
        cv2.imwrite(os.path.join(save_dir, f'{i}.png'), ndarray)
```

### 3.5 调整参数

选择新模型后，减少了过拟合情况的出现，训练后得到的正确率有所提高。之后，我们小组又进行了多轮的模型参数调整，最终得到了较为满意的模型预测正确率。调整参数和得到的正确率结果，如下图所示：

```python
model.prepare(paddle.optimizer.Adam(learning_rate=0.0004, parameters=model.parameters()),
              paddle.nn.CrossEntropyLoss(),
              paddle.metric.Accuracy(topk=(1, 2)))  # 返回top1 top2 的准确率


model.fit(train_data=dataset(data_path=target_path, mode='train'),
          eval_data= dataset(data_path=target_path, mode='eval'),
          epochs=10,
          batch_size=128,   #设太大有可能爆内存。
          save_dir="output/",
          save_freq=5,        #保存模型的频率，多少个 epoch 保存一次模型
          log_freq=10,     #日志打印的频率，多少个 step 打印一次日志
          shuffle=True)
```

```
The loss value printed in the log is the current step, and the metric is the avera
Epoch 1/10

/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/nn/laye
  warnings.warn(

step  10/227 - loss: 0.7201 - acc_top1: 0.5687 - acc_top2: 0.6453 - 288ms/step
step  20/227 - loss: 0.1883 - acc_top1: 0.7055 - acc_top2: 0.7656 - 275ms/step
step  30/227 - loss: 0.5190 - acc_top1: 0.7688 - acc_top2: 0.8224 - 262ms/step
step  40/227 - loss: 0.3834 - acc_top1: 0.8090 - acc_top2: 0.8543 - 262ms/step
step  50/227 - loss: 0.4645 - acc_top1: 0.8344 - acc_top2: 0.8750 - 254ms/step
step  60/227 - loss: 0.1161 - acc_top1: 0.8521 - acc_top2: 0.8896 - 254ms/step
step  70/227 - loss: 0.0965 - acc_top1: 0.8683 - acc_top2: 0.9036 - 253ms/step
step  80/227 - loss: 0.2062 - acc_top1: 0.8814 - acc_top2: 0.9137 - 254ms/step
step  90/227 - loss: 0.1812 - acc_top1: 0.8910 - acc_top2: 0.9208 - 253ms/step
step 100/227 - loss: 0.1407 - acc_top1: 0.8984 - acc_top2: 0.9275 - 253ms/step
step 110/227 - loss: 0.0618 - acc_top1: 0.9055 - acc_top2: 0.9331 - 252ms/step
step 120/227 - loss: 0.0412 - acc_top1: 0.9117 - acc_top2: 0.9382 - 252ms/step
step 130/227 - loss: 0.0609 - acc_top1: 0.9166 - acc_top2: 0.9422 - 253ms/step
step 140/227 - loss: 0.0464 - acc_top1: 0.9208 - acc_top2: 0.9458 - 253ms/step
step 150/227 - loss: 0.1034 - acc_top1: 0.9243 - acc_top2: 0.9487 - 254ms/step
step 160/227 - loss: 0.0432 - acc_top1: 0.9276 - acc_top2: 0.9515 - 255ms/step
step 170/227 - loss: 0.0498 - acc_top1: 0.9313 - acc_top2: 0.9541 - 253ms/step
step 180/227 - loss: 0.1986 - acc_top1: 0.9340 - acc_top2: 0.9562 - 252ms/step
step 190/227 - loss: 0.1888 - acc_top1: 0.9356 - acc_top2: 0.9576 - 252ms/step
step 200/227 - loss: 0.0757 - acc_top1: 0.9376 - acc_top2: 0.9592 - 251ms/step
step 210/227 - loss: 0.0682 - acc_top1: 0.9395 - acc_top2: 0.9610 - 251ms/step
step 220/227 - loss: 0.1937 - acc_top1: 0.9408 - acc_top2: 0.9622 - 251ms/step
step 227/227 - loss: 0.1649 - acc_top1: 0.9417 - acc_top2: 0.9629 - 250ms/step
save checkpoint at /home/aistudio/output/0
```

在调参的过程中，我们将多个轮次不同参数的结果进行了记录，如下图所示：

| Learning Rate | Epochs | Batch_Size | Loss | Acc | |
|---|---|---|---|---|---|
| 0.01 | 10 | 32 | 0.65 | 0.7544 | |
| 0.01 | 15 | 32 | 0.6 | 0.7701 | |
| 0.01 | 20 | 32 | 0.25 | 0.7852 | |
| 0.01 | 20 | 64 | 0.26 | 0.7892 | |
| 0.01 | 20 | 128 | 0.216 | 0.7885 | 此时运行时间上升，正确率并未提高多少 |
| 0.005 | 10 | 32 | 0.194 | 0.8186 | |
| 0.005 | 15 | 64 | 0.15 | 0.8624 | |
| 0.0001 | 10 | 32 | 0.23 | 0.7265 | 在学习率较低的情况下，epochs只能保持较低的数值才能保证运行效率 |
| 0.0005 | 10 | 32 | 0.086 | 0.9013 | |
| 0.0005 | 10 | 64 | 0.023 | 0.9513 | |
| 0.0004 | 15 | 128 | 0.003 | 0.9954 | 但是模型训练的时间有点长，考虑缩小一下epochs |
| 0.0004 | 10 | 128 | 0.0031 | 0.9912 | |

## 4 项目代码

```
import os
import zipfile
import json
import sys
import numpy as np
from PIL import Image
import random
import paddle
from paddle.io import Dataset

'''
参数配置
'''
train_parameters = {
    "input_size": [1, 20, 20],                             #输入图片的shape
    "class_dim": -1,                                       #分类数
    "src_path":"data/data275676/characterData.zip",        #原始数据集路径
    "target_path":"/home/aistudio/data/dataset",          #要解压的路径
    "train_list_path": "/home/aistudio/data/train_data.txt",
#train_data.txt路径
    "eval_list_path": "/home/aistudio/data/val_data.txt",
#eval_data.txt路径
    "label_dict":{},                                       #标签字典
    "readme_path": "/home/aistudio/data/readme.json",     #readme.json路径
}

src_path=train_parameters['src_path']
target_path=train_parameters['target_path']
train_list_path=train_parameters['train_list_path']
eval_list_path=train_parameters['eval_list_path']

def unzip_data(src_path,target_path):
    '''
    解压原始数据集，将src_path路径下的zip包解压至data/dataset目录下
```

```
    '''
    if(not os.path.isdir(target_path)):
        z = zipfile.ZipFile(src_path, 'r')
        z.extractall(path=target_path)
        z.close()
    else:
        print("文件已解压")

def get_data_list(target_path,train_list_path,eval_list_path):
    '''
    生成数据列表
    '''
    #存放所有类别的信息
    class_detail = []
    #获取所有类别保存的文件夹名称
    data_list_path=target_path
    class_dirs = os.listdir(data_list_path)
    if '__MACOSX' in class_dirs:
        class_dirs.remove('__MACOSX')
    # #总的图像数量
    all_class_images = 0
    # #存放类别标签
    class_label=0
    # #存放类别数目
    class_dim = 0
    # #存储要写进eval.txt和train.txt中的内容
    trainer_list=[]
    eval_list=[]
    #读取每个类别
    for class_dir in class_dirs:
        if class_dir != ".DS_Store":
            class_dim += 1
            #每个类别的信息
            class_detail_list = {}
            eval_sum = 0
            trainer_sum = 0
            #统计每个类别有多少张图片
            class_sum = 0
            #获取类别路径
            path = os.path.join(data_list_path,class_dir)
            # print(path)
            # 获取所有图片
            img_paths = os.listdir(path)
```

```
            for img_path in img_paths:                                  # 遍历文件夹下
的每个图片
                if img_path =='.DS_Store':
                    continue
                name_path = os.path.join(path,img_path)                 # 每张图
片的路径
                if class_sum % 10 == 0:                                 # 每10张图片取
一个做验证数据
                    eval_sum += 1                                       # eval_sum为测
试数据的数目
                    eval_list.append(name_path + "\t%d" % class_label + "\n")
                else:
                    trainer_sum += 1
                    trainer_list.append(name_path + "\t%d" % class_label +
"\n")#trainer_sum测试数据的数目
                class_sum += 1                                          #每类图片的数
目
                all_class_images += 1                                   #所有类图片的
数目

            # 说明的json文件的class_detail数据
            class_detail_list['class_name'] = class_dir             #类别名称
            class_detail_list['class_label'] = class_label          #类别标签
            class_detail_list['class_eval_images'] = eval_sum       #该类数据的测试集
数目
            class_detail_list['class_trainer_images'] = trainer_sum #该类数据的训练集
数目
            class_detail.append(class_detail_list)
            #初始化标签列表
            train_parameters['label_dict'][str(class_label)] = class_dir
            class_label += 1

    #初始化分类数
    train_parameters['class_dim'] = class_dim
    print(train_parameters)
    #乱序
    random.shuffle(eval_list)
    with open(eval_list_path, 'a') as f:
        for eval_image in eval_list:
            f.write(eval_image)
    #乱序
    random.shuffle(trainer_list)
    with open(train_list_path, 'a') as f2:
```

```
        for train_image in trainer_list:
            f2.write(train_image)

    # 说明的json文件信息
    readjson = {}
    readjson['all_class_name'] = data_list_path                  #文件父目录
    readjson['all_class_images'] = all_class_images
    readjson['class_detail'] = class_detail
    jsons = json.dumps(readjson, sort_keys=True, indent=4, separators=(',', ': '))
    with open(train_parameters['readme_path'],'w') as f:
        f.write(jsons)
    print ('生成数据列表完成！')

unzip_data(src_path,target_path)
with open(train_list_path, 'w') as f:
    f.seek(0)
    f.truncate()
with open(eval_list_path, 'w') as f:
    f.seek(0)
    f.truncate()

get_data_list(target_path,train_list_path,eval_list_path)
```

```
文件已解压
{'input_size': [1, 20, 20], 'class_dim': 65, 'src_path':
'data/data275676/characterData.zip', 'target_path': '/home/aistudio/data/dataset',
'train_list_path': '/home/aistudio/data/train_data.txt', 'eval_list_path':
'/home/aistudio/data/val_data.txt', 'label_dict': {'0': '5', '1': 'hei', '2': 'jing',
'3': 'J', '4': 'P', '5': '4', '6': 'L', '7': 'N', '8': 'lu', '9': 'G', '10': 'jin',
'11': 'zhe', '12': 'hu', '13': 'meng', '14': 'cuan', '15': 'Q', '16': 'yu1', '17':
'Z', '18': 'wan', '19': 'qing', '20': 'e1', '21': 'K', '22': 'V', '23': 'Y', '24':
'7', '25': 'E', '26': 'liao', '27': 'B', '28': 'zang', '29': '3', '30': 'gan1', '31':
'W', '32': 'U', '33': '0', '34': 'T', '35': '6', '36': 'su', '37': 'M', '38': '1',
'39': 'yue', '40': 'gan', '41': 'ji', '42': 'qiong', '43': 'R', '44': 'H', '45': 'S',
'46': 'X', '47': 'F', '48': 'D', '49': 'yun', '50': 'yu', '51': 'jl', '52': '2', '53':
'shan', '54': 'gui1', '55': 'C', '56': '8', '57': 'ning', '58': '9', '59': 'gui',
'60': 'sx', '61': 'min',
生成数据列表完成！
```

```
import random
class dataset(Dataset):
    def __init__(self, data_path, mode='train'):
        """
        数据读取器
```

```
        :param data_path: 数据集所在路径
        :param mode: train or eval
        """
        super().__init__()
        self.data_path = data_path
        self.img_paths = []
        self.labels = []

        if mode == 'train':
            filename = "train_data.txt"
        else:
            filename = "val_data.txt"

        with open(os.path.join("/home/aistudio/data/", filename), "r", encoding="utf-
8") as f:
            self.info = f.readlines()
        for img_info in self.info:
            img_path, label = img_info.strip().split('\t')
            if os.path.isfile(img_path):
                self.img_paths.append(img_path)
                self.labels.append(int(label))




    def __getitem__(self, index):
        """
        获取一组数据
        :param index: 文件索引号
        :return:
        """
        # 第一步打开图像文件并获取label值
        img_path = self.img_paths[index]
        if not os.path.isfile(img_path):
            raise FileNotFoundError(f"File not found or is not a file: {img_path}")

        img = Image.open(img_path)
        if img.mode != 'RGB':
            img = img.convert('RGB')
        img = img.resize((224, 224), Image.BILINEAR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
        img = np.array(img).astype('float32')
        img = img.transpose((2, 0, 1)) / 255
        label = self.labels[index]
        label = np.array([label], dtype="int64")
        return img, label


    def print_sample(self, index: int = 0):
        print("文件名", self.img_paths[index], "\t标签值", self.labels[index])

    def __len__(self):
        return len(self.img_paths)

class_number = train_parameters['class_dim']

network = paddle.vision.models.resnet34(num_classes=class_number,pretrained=True)
model = paddle.Model(network)
model.summary((-1, 3, 224, 224))  #input image size: 3*224*224
W0604 10:23:06.804745 134631 gpu_resources.cc:119] Please NOTE: device: 0, GPU Compute
Capability: 7.0, Driver API Version: 12.0, Runtime API Version: 11.8
W0604 10:23:06.806103 134631 gpu_resources.cc:164] device: 0, cuDNN Version: 8.9.
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-
packages/paddle/nn/layer/layers.py:2084: UserWarning: Skip loading for fc.weight.
fc.weight receives a shape [512, 1000], but the expected shape is [512, 65].
  warnings.warn(f"Skip loading for {key}. " + str(err))
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-
packages/paddle/nn/layer/layers.py:2084: UserWarning: Skip loading for fc.bias.
fc.bias receives a shape [1000], but the expected shape is [65].
  warnings.warn(f"Skip loading for {key}. " + str(err))

 model.prepare(paddle.optimizer.Adam(learning_rate=0.0005,
parameters=model.parameters()),
              paddle.nn.CrossEntropyLoss(),
              paddle.metric.Accuracy(topk=(1, 2)))  # 返回top1 top2 的准确率



model.fit(train_data=dataset(data_path=target_path, mode='train'),
          eval_data= dataset(data_path=target_path, mode='eval'),
          epochs=10,
          batch_size=128,    #设太大有可能爆内存。
          save_dir="output/",
          save_freq=5,        #保存模型的频率，多少个 epoch 保存一次模型
```

```
          log_freq=10,      #日志打印的频率，多少个 step 打印一次日志
          shuffle=True)

#将标签进行转换
print('Label:',train_parameters['label_dict'])
match =
{'A':'A','B':'B','C':'C','D':'D','E':'E','F':'F','G':'G','H':'H','I':'I','J':'J','K':'
K','L':'L','M':'M','N':'N',

'O':'O','P':'P','Q':'Q','R':'R','S':'S','T':'T','U':'U','V':'V','W':'W','X':'X','Y':'Y
','Z':'Z',

'yun':'云','cuan':'川','hei':'黑','zhe':'浙','ning':'宁','jin':'津','gan':'赣','hu':'
沪','liao':'辽','jl':'吉','qing':'青','zang':'藏',

'e1':'鄂','meng':'蒙','gan1':'甘','qiong':'琼','shan':'陕','min':'闽','su':'苏','xin':
'新','wan':'皖','jing':'京','xiang':'湘','gui':'贵',
        'yu1':'渝','yu':'豫','ji':'冀','yue':'粤','gui1':'桂','sx':'晋','lu':'鲁',

'0':'0','1':'1','2':'2','3':'3','4':'4','5':'5','6':'6','7':'7','8':'8','9':'9'}
L = 0
LABEL ={}
for V in train_parameters['label_dict'].values():
    LABEL[str(L)] = match[V]
    L += 1
print(LABEL)
#print(len(LABEL))
print(LABEL['0'])

import cv2
import numpy as np
import os
import paddle
from PIL import Image

def process_license_plate(image_path, save_dir):
    # 读取图像
    license_plate = cv2.imread(image_path)
    gray_plate = cv2.cvtColor(license_plate, cv2.COLOR_RGB2GRAY)
    ret, binary_plate = cv2.threshold(gray_plate, 175, 255, cv2.THRESH_BINARY) # ret:
阈值，binary_plate：根据阈值处理后的图像数据

    # 按列统计像素分布
```

```
    result = []
    for col in range(binary_plate.shape[1]):
        result.append(0)
        for row in range(binary_plate.shape[0]):
            result[col] += binary_plate[row][col] / 255

    # 行上的像素
    width = binary_plate.shape[0]
    ch_width = width / 8

    # 记录车牌中字符的位置
    character_dict = {}
    num = 0
    i = 0
    while i < len(result):
        if result[i] == 0:
            i += 1
        else:
            index = i + 1
            while index < len(result) and result[index] != 0:
                index += 1
            character_dict[num] = [i, index - 1]
            num += 1
            i = index

    # 将每个字符填充，并存储
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    for i in range(8):
        if i == 2:
            continue
        padding = (170 - (character_dict[i][1] - character_dict[i][0])) / 2
        ndarray = np.pad(binary_plate[:, character_dict[i][0]:character_dict[i][1]],
((0, 0), (int(padding), int(padding))), 'constant', constant_values=(0, 0))
        ndarray = cv2.resize(ndarray, (20, 20))
        cv2.imwrite(os.path.join(save_dir, f'{i}.png'), ndarray)
        # 如果需要将字符添加到一个列表中，可取消注释以下行
        # characters.append(ndarray)

def load_image(path):
    img = Image.open(path).convert('L')
    img = np.array(img).astype('float32')
```

```
    img = img[np.newaxis, ] / 255.0
    return img

# 遍历 infer 文件夹中的每张图片
infer_dir = '/home/aistudio/infer/'
for image_filename in os.listdir(infer_dir):
    if (image_filename.endswith('.png') or image_filename.endswith('.jpg')):  # 确保是
图片文件
        image_path = os.path.join(infer_dir, image_filename)
        save_dir = os.path.join('/home/aistudio/infer/out',
os.path.splitext(image_filename)[0])
        process_license_plate(image_path, save_dir)

def predict_character_images(image_folder):
    all_path = []
    for image_filename in os.listdir(image_folder):
        #print(image_filename)
        img_cut_path = os.path.join(image_folder,image_filename)
#/home/aistudio/infer/out/car2
        char_img_paths = []
        for i in range(8):
            if i == 2:
                continue
            char_img_paths.append(os.path.join(img_cut_path, f"{i}.png"))
        all_path.append(char_img_paths)

    return all_path

image_folder = '/home/aistudio/infer/out/'
paths = []
paths = predict_character_images(image_folder)
#print(paths[1])
#print(paths[0])
print(len(paths))

import matplotlib.pyplot as plt

class InferDataset(Dataset):
    def __init__(self, img_path=None):
        """
        数据读取Reader(推理)
        :param img_path: 推理单张图片，这是为一张图片建立的dataset
        """
```

```
        super().__init__()
        if img_path:
            self.img_paths = [img_path]
        else:
            raise Exception("请指定需要预测对应图片路径")



    def __getitem__(self, index):
        # 获取图像路径
        img_path = self.img_paths[index]
        # 使用Pillow来读取图像数据并转成Numpy格式
        img = Image.open(img_path)
        if img.mode != 'RGB':
            img = img.convert('RGB')
        img = img.resize((224, 224), Image.BILINEAR)
        img = np.array(img).astype('float32')
        img = img.transpose((2, 0, 1)) / 255  # HWC to CHW 并像素归一化
        return img

    def __len__(self):
        return len(self.img_paths)

#开始预测
network = paddle.vision.models.resnet34(num_classes=class_number,pretrained=True)
# 实例化推理模型
model = paddle.Model(network)
# 读取刚刚训练好的参数
model.load("./output/final")
# 初始化模型
model.prepare()

result_list = []

for i in range(len(paths)):
    character_list = []
    for j in range(7):
        img_cut_path_infer = paths[i][j]
        infer_data = InferDataset(img_cut_path_infer)
        result = model.predict(test_data=infer_data)
        #print('result: Label',result)
        #print('result: Label',np.argmax(result))
        character = str(np.argmax(result))
```

```
            print(LABEL[character])
            character_list.append(LABEL[character])
        print('\n')
        result_list.append(character_list)

# 对数据集进行模型预测
for i in range(40):
    print(paths[i])
    print(result_list[i])
```

## 5 运行结果

模型训练完后，我们首先对单张图片进行了测试，得到的结果与测试图片相同。并进行了一些多张图片的测试，得到结果的正确率也极高。

```
Predict begin...
step 1/1 [==============================] - 113ms/step
Predict samples: 1
鲁
Predict begin...
step 1/1 [==============================] - 16ms/step
Predict samples: 1
A
Predict begin...
step 1/1 [==============================] - 15ms/step
Predict samples: 1
6
Predict begin...
step 1/1 [==============================] - 16ms/step
Predict samples: 1
8
Predict begin...
step 1/1 [==============================] - 14ms/step
Predict samples: 1
6
Predict begin...
step 1/1 [==============================] - 14ms/step
Predict samples: 1
E
Predict begin...
step 1/1 [==============================] - 110ms/step
运行时长: 1.855秒   结束时间: 2024-06-03 20:08:49
```

```
print(result_list[1])
```

```
['/home/aistudio/infer/out/4/0.png', '/home/aistudio/infer/out/4/1.png', '/home/aistudio/infer/out/4/3.png', '/home/aistudio/infer/out/4/4.png', '/home/aistudio/infer/out/4/5.png', '/home/aistudio/infer/out/4/6.png', '/home/aistudio/infer/out/4/7.png']
['苏', 'G', 'J', 'Y', '7', '7', 'S']
['/home/aistudio/infer/out/3/0.png', '/home/aistudio/infer/out/3/1.png', '/home/aistudio/infer/out/3/3.png', '/home/aistudio/infer/out/3/4.png', '/home/aistudio/infer/out/3/5.png', '/home/aistudio/infer/out/3/6.png', '/home/aistudio/infer/out/3/7.png']
['闽', 'A', 'S', '5', 'C', 'G', 'E']
['/home/aistudio/infer/out/1/0.png', '/home/aistudio/infer/out/1/1.png', '/home/aistudio/infer/out/1/3.png', '/home/aistudio/infer/out/1/4.png', '/home/aistudio/infer/out/1/5.png', '/home/aistudio/infer/out/1/6.png', '/home/aistudio/infer/out/1/7.png']
['闽', 'C', '2', '5', 'A', '8', 'C']
['/home/aistudio/infer/out/5/0.png', '/home/aistudio/infer/out/5/1.png', '/home/aistudio/infer/out/5/3.png', '/home/aistudio/infer/out/5/4.png', '/home/aistudio/infer/out/5/5.png', '/home/aistudio/infer/out/5/6.png', '/home/aistudio/infer/out/5/7.png']
['湘', 'A', 'R', '1', 'L', 'J', 'R']
['/home/aistudio/infer/out/21/0.png', '/home/aistudio/infer/out/21/1.png', '/home/aistudio/infer/out/21/3.png', '/home/aistudio/infer/out/21/4.png', '/home/aistudio/infer/out/21/5.png', '/home/aistudio/infer/out/21/6.png', '/home/aistudio/infer/out/21/7.png']
['京', 'F', 'F', '9', 'M', '1', '4']
['/home/aistudio/infer/out/26/0.png', '/home/aistudio/infer/out/26/1.png', '/home/aistudio/infer/out/26/3.png', '/home/aistudio/infer/out/26/4.png', '/home/aistudio/infer/out/26/5.png', '/home/aistudio/infer/out/26/6.png', '/home/aistudio/infer/out/26/7.png']
['桂', 'E', 'S', '5', '9', '8', '6']
['/home/aistudio/infer/out/28/0.png', '/home/aistudio/infer/out/28/1.png', '/home/aistudio/infer/out/28/3.png', '/home/aistudio/infer/out/28/4.png', '/home/aistudio/infer/out/28/5.png', '/home/aistudio/infer/out/28/6.png', '/home/aistudio/infer/out/28/7.png']
['贵', 'C', '5', '4', '8', '7', '5']
['/home/aistudio/infer/out/10/0.png', '/home/aistudio/infer/out/10/1.png', '/home/aistudio/infer/out/10/3.png', '/home/aistudio/infer/out/10/4.png', '/home/aistudio/infer/out/10/5.png', '/home/aistudio/infer/out/10/6.png', '/home/aistudio/infer/out/10/7.png']
['赣', 'H', 'T', '3', 'B', 'U', 'Q']
['/home/aistudio/infer/out/24/0.png', '/home/aistudio/infer/out/24/1.png', '/home/aistudio/infer/out/24/3.png', '/home/aistudio/infer/out/24/4.png', '/home/aistudio/infer/out/24/5.png', '/home/aistudio/infer/out/24/6.png', '/home/aistudio/infer/out/24/7.png']
['藏', 'D', '9', '4', '2', '1', '6']
['/home/aistudio/infer/out/19/0.png', '/home/aistudio/infer/out/19/1.png', '/home/aistudio/infer/out/19/3.png', '/home/aistudio/infer/out/19/4.png', '/home/aistudio/infer/out/19/5.png', '/home/aistudio/infer/out/19/6.png', '/home/aistudio/infer/out/19/7.png']
['闽', 'D', '3', '5', 'J', 'H', 'B']
['/home/aistudio/infer/out/23/0.png', '/home/aistudio/infer/out/23/1.png', '/home/aistudio/infer/out/23/3.png', '/home/aistudio/infer/out/23/4.png', '/home/aistudio/infer/out/23/5.png', '/home/aistudio/infer/out/23/6.png', '/home/aistudio/infer/out/23/7.png']
['辽', 'J', 'S', '0', '7', '6', '8']
```

因此我们决定采用数据集的方式对模型进行测试和调整，但是在网络上寻找未发现使用整张车牌照片作为输入的相关数据集。因此我们小组成员选择在网络上手动查找 140 张左右的图片，制作了两个测试数据集分别进行测试。

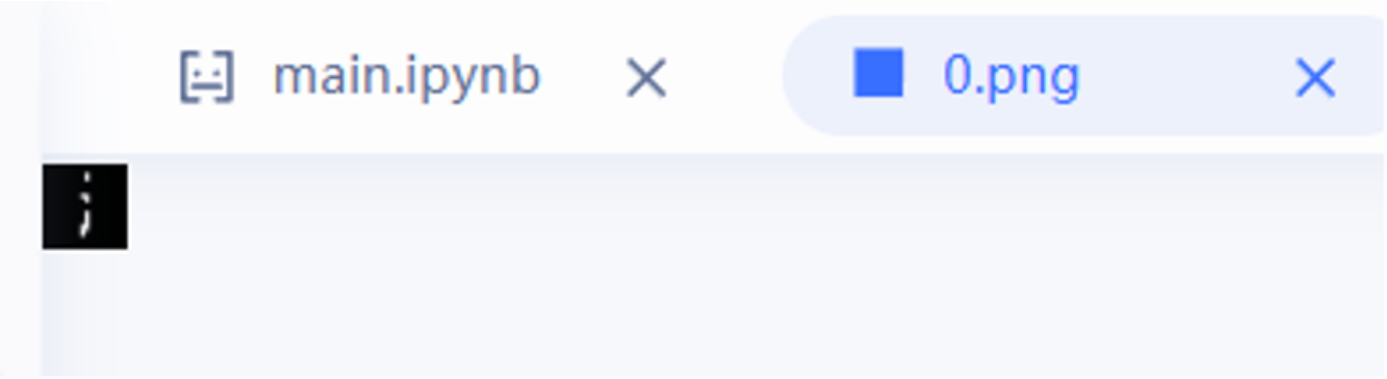
最终得到的正确率在 96% 左右，符合我们在模型训练前的预期，实现了对应的功能。

---

## 6 改进方向

经过检查预测错误的数据，我们发现，我们的图像预处理方法在汉字 “渝” 的处理上会出现问题。由于 “渝” 是左右结构的汉字，因此按列进行像素块的分割时，会将左边的三点水和右边的俞分别分开形成两个

字，出现如下图所示的处理情况，导致预测的结果出现错误。




因此，我们还可以在 “渝” 这个汉字的处理上进行一定的修改优化。可以在分割出多一张照片的情况下，将左右两个汉字合在一起分类成 “渝” ，这样就可以对所有的省份进行正确的分类，避免发生错误。

---

## 7 组内分工

- 杜佳钰： 测试数据集查找，图像预处理，撰写项目报告
- 鄢志轩：训练数据集查找，模型训练调试，汇报展示
- 温恺宣：测试数据集查找，模型调优记录

---