**Archive**

**一、中文OCR预训练模型**

**建模文档**

1. **超参数设置**

BATCH\_SIZE = 32

NUM\_EPOCHS = 10

LEARNING\_RATE = 0.0001

RANDOM\_SEED = 42

1. **数据介绍**

根据文字以及背景图片生成数据集。首先通过drawText生成对应fontpath字体的文字图片，然后以一定的概率通过AddBackgroud把文字放到background\_img\_paths中的背景上面。训练集：验证集 = 8：2

1. **数据预处理方法（数据裁剪、增广等）**

Rotate 不超过60°的随机旋转

CenterCrop(32,32), 中心裁剪为32 \* 32大小

AddBackgroud 增加背景

HueSaturationValue(p=0.3), 随机色调饱和度值

RandomBrightnessContrast(p=0.3), 随机亮度对比度

ToGray(p=0.3), 0.3的概率随机转化为灰度图

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

1. **建模流程、模型介绍**
   1. 迁移模型

Resnet34 的4-9层

4.2下游模型更改

1-3层为torch.nn.Conv2d(3,64,kernel\_size=(1,1),stride=(1,1),padding=(0,0),bias=False),

torch.nn.BatchNorm2d(64),

torch.nn.ReLU(inplace=True)

Fc层为 torch.nn.Linear(in\_features=512, out\_features=n\_classes, bias = True)

4.3模型总架构

----------------------------------------------------------------

Layer (type) Output Shape Param #

================================================================

Conv2d-1 [-1, 64, 32, 32] 192

BatchNorm2d-2 [-1, 64, 32, 32] 128

ReLU-3 [-1, 64, 32, 32] 0

Conv2d-4 [-1, 64, 32, 32] 36,864

BatchNorm2d-5 [-1, 64, 32, 32] 128

ReLU-6 [-1, 64, 32, 32] 0

Conv2d-7 [-1, 64, 32, 32] 36,864

BatchNorm2d-8 [-1, 64, 32, 32] 128

ReLU-9 [-1, 64, 32, 32] 0

BasicBlock-10 [-1, 64, 32, 32] 0

Conv2d-11 [-1, 64, 32, 32] 36,864

BatchNorm2d-12 [-1, 64, 32, 32] 128

ReLU-13 [-1, 64, 32, 32] 0

Conv2d-14 [-1, 64, 32, 32] 36,864

BatchNorm2d-15 [-1, 64, 32, 32] 128

ReLU-16 [-1, 64, 32, 32] 0

BasicBlock-17 [-1, 64, 32, 32] 0

Conv2d-18 [-1, 64, 32, 32] 36,864

BatchNorm2d-19 [-1, 64, 32, 32] 128

ReLU-20 [-1, 64, 32, 32] 0

Conv2d-21 [-1, 64, 32, 32] 36,864

BatchNorm2d-22 [-1, 64, 32, 32] 128

ReLU-23 [-1, 64, 32, 32] 0

BasicBlock-24 [-1, 64, 32, 32] 0

Conv2d-25 [-1, 128, 16, 16] 73,728

BatchNorm2d-26 [-1, 128, 16, 16] 256

ReLU-27 [-1, 128, 16, 16] 0

Conv2d-28 [-1, 128, 16, 16] 147,456

BatchNorm2d-29 [-1, 128, 16, 16] 256

Conv2d-30 [-1, 128, 16, 16] 8,192

BatchNorm2d-31 [-1, 128, 16, 16] 256

ReLU-32 [-1, 128, 16, 16] 0

BasicBlock-33 [-1, 128, 16, 16] 0

Conv2d-34 [-1, 128, 16, 16] 147,456

BatchNorm2d-35 [-1, 128, 16, 16] 256

ReLU-36 [-1, 128, 16, 16] 0

Conv2d-37 [-1, 128, 16, 16] 147,456

BatchNorm2d-38 [-1, 128, 16, 16] 256

ReLU-39 [-1, 128, 16, 16] 0

BasicBlock-40 [-1, 128, 16, 16] 0

Conv2d-41 [-1, 128, 16, 16] 147,456

BatchNorm2d-42 [-1, 128, 16, 16] 256

ReLU-43 [-1, 128, 16, 16] 0

Conv2d-44 [-1, 128, 16, 16] 147,456

BatchNorm2d-45 [-1, 128, 16, 16] 256

ReLU-46 [-1, 128, 16, 16] 0

BasicBlock-47 [-1, 128, 16, 16] 0

Conv2d-48 [-1, 128, 16, 16] 147,456

BatchNorm2d-49 [-1, 128, 16, 16] 256

ReLU-50 [-1, 128, 16, 16] 0

Conv2d-51 [-1, 128, 16, 16] 147,456

BatchNorm2d-52 [-1, 128, 16, 16] 256

ReLU-53 [-1, 128, 16, 16] 0

BasicBlock-54 [-1, 128, 16, 16] 0

Conv2d-55 [-1, 256, 8, 8] 294,912

BatchNorm2d-56 [-1, 256, 8, 8] 512

ReLU-57 [-1, 256, 8, 8] 0

Conv2d-58 [-1, 256, 8, 8] 589,824

BatchNorm2d-59 [-1, 256, 8, 8] 512

Conv2d-60 [-1, 256, 8, 8] 32,768

BatchNorm2d-61 [-1, 256, 8, 8] 512

ReLU-62 [-1, 256, 8, 8] 0

BasicBlock-63 [-1, 256, 8, 8] 0

Conv2d-64 [-1, 256, 8, 8] 589,824

BatchNorm2d-65 [-1, 256, 8, 8] 512

ReLU-66 [-1, 256, 8, 8] 0

Conv2d-67 [-1, 256, 8, 8] 589,824

BatchNorm2d-68 [-1, 256, 8, 8] 512

ReLU-69 [-1, 256, 8, 8] 0

BasicBlock-70 [-1, 256, 8, 8] 0

Conv2d-71 [-1, 256, 8, 8] 589,824

BatchNorm2d-72 [-1, 256, 8, 8] 512

ReLU-73 [-1, 256, 8, 8] 0

Conv2d-74 [-1, 256, 8, 8] 589,824

BatchNorm2d-75 [-1, 256, 8, 8] 512

ReLU-76 [-1, 256, 8, 8] 0

BasicBlock-77 [-1, 256, 8, 8] 0

Conv2d-78 [-1, 256, 8, 8] 589,824

BatchNorm2d-79 [-1, 256, 8, 8] 512

ReLU-80 [-1, 256, 8, 8] 0

Conv2d-81 [-1, 256, 8, 8] 589,824

BatchNorm2d-82 [-1, 256, 8, 8] 512

ReLU-83 [-1, 256, 8, 8] 0

BasicBlock-84 [-1, 256, 8, 8] 0

Conv2d-85 [-1, 256, 8, 8] 589,824

BatchNorm2d-86 [-1, 256, 8, 8] 512

ReLU-87 [-1, 256, 8, 8] 0

Conv2d-88 [-1, 256, 8, 8] 589,824

BatchNorm2d-89 [-1, 256, 8, 8] 512

ReLU-90 [-1, 256, 8, 8] 0

BasicBlock-91 [-1, 256, 8, 8] 0

Conv2d-92 [-1, 256, 8, 8] 589,824

BatchNorm2d-93 [-1, 256, 8, 8] 512

ReLU-94 [-1, 256, 8, 8] 0

Conv2d-95 [-1, 256, 8, 8] 589,824

BatchNorm2d-96 [-1, 256, 8, 8] 512

ReLU-97 [-1, 256, 8, 8] 0

BasicBlock-98 [-1, 256, 8, 8] 0

Conv2d-99 [-1, 512, 4, 4] 1,179,648

BatchNorm2d-100 [-1, 512, 4, 4] 1,024

ReLU-101 [-1, 512, 4, 4] 0

Conv2d-102 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-103 [-1, 512, 4, 4] 1,024

Conv2d-104 [-1, 512, 4, 4] 131,072

BatchNorm2d-105 [-1, 512, 4, 4] 1,024

ReLU-106 [-1, 512, 4, 4] 0

BasicBlock-107 [-1, 512, 4, 4] 0

Conv2d-108 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-109 [-1, 512, 4, 4] 1,024

ReLU-110 [-1, 512, 4, 4] 0

Conv2d-111 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-112 [-1, 512, 4, 4] 1,024

ReLU-113 [-1, 512, 4, 4] 0

BasicBlock-114 [-1, 512, 4, 4] 0

Conv2d-115 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-116 [-1, 512, 4, 4] 1,024

ReLU-117 [-1, 512, 4, 4] 0

Conv2d-118 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-119 [-1, 512, 4, 4] 1,024

ReLU-120 [-1, 512, 4, 4] 0

BasicBlock-121 [-1, 512, 4, 4] 0

AdaptiveAvgPool2d-122 [-1, 512, 1, 1] 0

Linear-123 [-1, 3755] 1,926,315

================================================================

Total params: 23,201,771

Trainable params: 23,201,771

Non-trainable params: 0

----------------------------------------------------------------

Input size (MB): 0.01

Forward/backward pass size (MB): 26.47

Params size (MB): 88.51

Estimated Total Size (MB): 114.99

----------------------------------------------------------------

1. **训练方法**

train\_model

Lr\_schedule: ReduceLROnPlateau(

optimizer,

patience=1,

threshold=0.01,

factor = 0.2,

verbose=True,

mode="max")

metric\_func = accuracy

Loss ： cross\_entropy

Optimizer : Adam ：lr = LEARNING\_RATE

最好的权重：checkpoint =

torch.load("../models/ocr\_model\_checkpoint/best\_model.pt")

**Serving**

1. **后端框架介绍**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径pretrained\_predict

1. **POST ApI 设计**

POST接口调用PyTorch\_REST\_API\_URL + "pretrained\_predict"

1. **返回数据示例**

{'prediction': [{'label': '一', 'location': [38, 25]}, {'label': '旷', 'location': [73, 51]}, {'label': '吉', 'location': [95, 123]}, {'label': '一', 'location': [181, 44]}, {'label': '削', 'location': [224, 65]}, {'label': '范', 'location': [234, 115]}, {'label': '式', 'location': [265, 41]}], 'success': True}

1. **Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "pretrained\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/1.png","rb") as f:

base64\_data = base64.b64encode(f.read())

print(pretrained\_predict(base64\_data))

**二、天眼查OCR中文文字识别**

**建模文档**

1. **超参数设置**

BATCH\_SIZE = 16

NUM\_EPOCHS = 20

LEARNING\_RATE = 0.0001

RANDOM\_SEED = 42

1. **数据介绍**

自己标注的的数据集，在0和1文件夹下的图片，图片命名格式为char\_index。

此处在标注数据集之后要用代码检查标注错误的文件。训练集：验证集 = 9 ：1。

from pathlib import Path

root = Path("data")

for direc in root.glob("\*"):

if direc.is\_dir():

source = direc / "0"

target = direc / "1"

target\_set = set()

for t\_char in target.glob("\*.png"):

target\_set.add(t\_char.name[0])

for s\_char in source.glob("\*.png"):

if s\_char.name[0] not in target\_set:

print(direc.name)

print(s\_char.name[0])

1. **数据预处理方法（数据裁剪、增广等）**

把数据集扩充5倍，然后shuffle打乱，然后再通过transforms对图片进行预处理

Resize(36,36), 将输入图像调整为给定的高度和宽度

Rotate(60,interpolation = cv2.INTER\_LINEAR), 不超过60°的随机旋转

CenterCrop(32,32), 中心裁剪为32 \* 32大小

HueSaturationValue(p=0.3), 随机色调饱和度值

RandomBrightnessContrast(p=0.3), 随机亮度对比度

ToGray(p=0.3), 转化为灰度图

OpticalDistortion(distort\_limit=0.03, p = 0.3), 对图像进行光学畸变

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

1. **建模流程、模型介绍**

此处是直接加载的pretrained\_predict训练出来的best checkpoint ：

OCR/models/ocr\_pretrained\_model\_checkpoint/best\_model.pt

* 1. 迁移模型

Resnet34 的4-9层

4.2下游模型更改

1-3层为torch.nn.Conv2d(3,64,kernel\_size=(1,1),stride=(1,1),padding=(0,0),bias=False),

torch.nn.BatchNorm2d(64),

torch.nn.ReLU(inplace=True)

Fc层为 torch.nn.Linear(in\_features=512, out\_features=n\_classes, bias = True)

4.3模型总架构

----------------------------------------------------------------

Layer (type) Output Shape Param #

================================================================

Conv2d-1 [-1, 64, 32, 32] 192

BatchNorm2d-2 [-1, 64, 32, 32] 128

ReLU-3 [-1, 64, 32, 32] 0

Conv2d-4 [-1, 64, 32, 32] 36,864

BatchNorm2d-5 [-1, 64, 32, 32] 128

ReLU-6 [-1, 64, 32, 32] 0

Conv2d-7 [-1, 64, 32, 32] 36,864

BatchNorm2d-8 [-1, 64, 32, 32] 128

ReLU-9 [-1, 64, 32, 32] 0

BasicBlock-10 [-1, 64, 32, 32] 0

Conv2d-11 [-1, 64, 32, 32] 36,864

BatchNorm2d-12 [-1, 64, 32, 32] 128

ReLU-13 [-1, 64, 32, 32] 0

Conv2d-14 [-1, 64, 32, 32] 36,864

BatchNorm2d-15 [-1, 64, 32, 32] 128

ReLU-16 [-1, 64, 32, 32] 0

BasicBlock-17 [-1, 64, 32, 32] 0

Conv2d-18 [-1, 64, 32, 32] 36,864

BatchNorm2d-19 [-1, 64, 32, 32] 128

ReLU-20 [-1, 64, 32, 32] 0

Conv2d-21 [-1, 64, 32, 32] 36,864

BatchNorm2d-22 [-1, 64, 32, 32] 128

ReLU-23 [-1, 64, 32, 32] 0

BasicBlock-24 [-1, 64, 32, 32] 0

Conv2d-25 [-1, 128, 16, 16] 73,728

BatchNorm2d-26 [-1, 128, 16, 16] 256

ReLU-27 [-1, 128, 16, 16] 0

Conv2d-28 [-1, 128, 16, 16] 147,456

BatchNorm2d-29 [-1, 128, 16, 16] 256

Conv2d-30 [-1, 128, 16, 16] 8,192

BatchNorm2d-31 [-1, 128, 16, 16] 256

ReLU-32 [-1, 128, 16, 16] 0

BasicBlock-33 [-1, 128, 16, 16] 0

Conv2d-34 [-1, 128, 16, 16] 147,456

BatchNorm2d-35 [-1, 128, 16, 16] 256

ReLU-36 [-1, 128, 16, 16] 0

Conv2d-37 [-1, 128, 16, 16] 147,456

BatchNorm2d-38 [-1, 128, 16, 16] 256

ReLU-39 [-1, 128, 16, 16] 0

BasicBlock-40 [-1, 128, 16, 16] 0

Conv2d-41 [-1, 128, 16, 16] 147,456

BatchNorm2d-42 [-1, 128, 16, 16] 256

ReLU-43 [-1, 128, 16, 16] 0

Conv2d-44 [-1, 128, 16, 16] 147,456

BatchNorm2d-45 [-1, 128, 16, 16] 256

ReLU-46 [-1, 128, 16, 16] 0

BasicBlock-47 [-1, 128, 16, 16] 0

Conv2d-48 [-1, 128, 16, 16] 147,456

BatchNorm2d-49 [-1, 128, 16, 16] 256

ReLU-50 [-1, 128, 16, 16] 0

Conv2d-51 [-1, 128, 16, 16] 147,456

BatchNorm2d-52 [-1, 128, 16, 16] 256

ReLU-53 [-1, 128, 16, 16] 0

BasicBlock-54 [-1, 128, 16, 16] 0

Conv2d-55 [-1, 256, 8, 8] 294,912

BatchNorm2d-56 [-1, 256, 8, 8] 512

ReLU-57 [-1, 256, 8, 8] 0

Conv2d-58 [-1, 256, 8, 8] 589,824

BatchNorm2d-59 [-1, 256, 8, 8] 512

Conv2d-60 [-1, 256, 8, 8] 32,768

BatchNorm2d-61 [-1, 256, 8, 8] 512

ReLU-62 [-1, 256, 8, 8] 0

BasicBlock-63 [-1, 256, 8, 8] 0

Conv2d-64 [-1, 256, 8, 8] 589,824

BatchNorm2d-65 [-1, 256, 8, 8] 512

ReLU-66 [-1, 256, 8, 8] 0

Conv2d-67 [-1, 256, 8, 8] 589,824

BatchNorm2d-68 [-1, 256, 8, 8] 512

ReLU-69 [-1, 256, 8, 8] 0

BasicBlock-70 [-1, 256, 8, 8] 0

Conv2d-71 [-1, 256, 8, 8] 589,824

BatchNorm2d-72 [-1, 256, 8, 8] 512

ReLU-73 [-1, 256, 8, 8] 0

Conv2d-74 [-1, 256, 8, 8] 589,824

BatchNorm2d-75 [-1, 256, 8, 8] 512

ReLU-76 [-1, 256, 8, 8] 0

BasicBlock-77 [-1, 256, 8, 8] 0

Conv2d-78 [-1, 256, 8, 8] 589,824

BatchNorm2d-79 [-1, 256, 8, 8] 512

ReLU-80 [-1, 256, 8, 8] 0

Conv2d-81 [-1, 256, 8, 8] 589,824

BatchNorm2d-82 [-1, 256, 8, 8] 512

ReLU-83 [-1, 256, 8, 8] 0

BasicBlock-84 [-1, 256, 8, 8] 0

Conv2d-85 [-1, 256, 8, 8] 589,824

BatchNorm2d-86 [-1, 256, 8, 8] 512

ReLU-87 [-1, 256, 8, 8] 0

Conv2d-88 [-1, 256, 8, 8] 589,824

BatchNorm2d-89 [-1, 256, 8, 8] 512

ReLU-90 [-1, 256, 8, 8] 0

BasicBlock-91 [-1, 256, 8, 8] 0

Conv2d-92 [-1, 256, 8, 8] 589,824

BatchNorm2d-93 [-1, 256, 8, 8] 512

ReLU-94 [-1, 256, 8, 8] 0

Conv2d-95 [-1, 256, 8, 8] 589,824

BatchNorm2d-96 [-1, 256, 8, 8] 512

ReLU-97 [-1, 256, 8, 8] 0

BasicBlock-98 [-1, 256, 8, 8] 0

Conv2d-99 [-1, 512, 4, 4] 1,179,648

BatchNorm2d-100 [-1, 512, 4, 4] 1,024

ReLU-101 [-1, 512, 4, 4] 0

Conv2d-102 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-103 [-1, 512, 4, 4] 1,024

Conv2d-104 [-1, 512, 4, 4] 131,072

BatchNorm2d-105 [-1, 512, 4, 4] 1,024

ReLU-106 [-1, 512, 4, 4] 0

BasicBlock-107 [-1, 512, 4, 4] 0

Conv2d-108 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-109 [-1, 512, 4, 4] 1,024

ReLU-110 [-1, 512, 4, 4] 0

Conv2d-111 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-112 [-1, 512, 4, 4] 1,024

ReLU-113 [-1, 512, 4, 4] 0

BasicBlock-114 [-1, 512, 4, 4] 0

Conv2d-115 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-116 [-1, 512, 4, 4] 1,024

ReLU-117 [-1, 512, 4, 4] 0

Conv2d-118 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-119 [-1, 512, 4, 4] 1,024

ReLU-120 [-1, 512, 4, 4] 0

BasicBlock-121 [-1, 512, 4, 4] 0

AdaptiveAvgPool2d-122 [-1, 512, 1, 1] 0

Linear-123 [-1, 3755] 1,926,315

================================================================

Total params: 23,201,771

Trainable params: 23,201,771

Non-trainable params: 0

----------------------------------------------------------------

Input size (MB): 0.01

Forward/backward pass size (MB): 26.47

Params size (MB): 88.51

Estimated Total Size (MB): 114.99

----------------------------------------------------------------

1. **训练方法**

train\_model

Lr\_schedule: ReduceLROnPlateau(

optimizer,

patience=2,

threshold=0.01,

factor = 0.2,

verbose=True,

mode="max")

metric\_func = accuracy

Loss ： cross\_entropy

Optimizer : Adam ：lr = LEARNING\_RATE

Loss ： cross\_entropy

**Serving**

1. **后端框架介绍**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径tyc\_predict

1. **POST ApI 设计**

POST接口调用PyTorch\_REST\_API\_URL + "tyc\_predict"

1. **返回数据示例**

{'prediction': [{'label': '三', 'location': [38, 25]}, {'label': '解', 'location': [73, 51]}, {'label': '害', 'location': [95, 123]}, {'label': '乏', 'location': [181, 44]}, {'label': '勘', 'location': [224, 65]}, {'label': '潜', 'location': [234, 115]}, {'label': '谗', 'location': [265, 41]}], 'success': True}

**4.Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "tyc\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/1.png","rb") as f:

base64\_data = base64.b64encode(f.read())

**三、国家企业信息信用中心图片+中文OCR识别**

**建模文档**

1. **超参数设置**

**1.1图片内容为文字部分**

BATCH\_SIZE = 16

NUM\_EPOCHS = 10

LEARNING\_RATE = 0.0001

RANDOM\_SEED = 42

**1.2图片内容为实物部分**

BATCH\_SIZE = 16

NUM\_EPOCHS = 30

LEARNING\_RATE = 0.0001

RANDOM\_SEED = 42

1. **数据介绍**

**2.1图片内容为文字部分**

根据文字生成数据集。首先通过drawText生成对应font\_paths字体的文字图片，文字为黑色，背景为白色。

**2.2图片内容为实物部分**

1. 实际验证码图片，手工标记。
2. 用爬虫crawler.py爬取相应的类别的图片。
3. **数据预处理方法（数据裁剪、增广等）**

**3.1图片内容为文字部分**

3.1.1训练集

RandomRotation(30,fill = (255,255,255)), 随机旋转30°

CenterCrop(size = 32), 中心裁剪为32 \* 32

ColorJitter(contrast = 0.4, saturation = 0.4), 对颜色的数据增强：图像亮度、饱和度、对比度变化

RandomGrayscale(p=1.0), 将图像转换为灰度图像

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

3.1.2测试集

Resize(32), 将大小缩放至32 \* 32

RandomGrayscale(p=1.0), 将图像转换为灰度图像

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

**3.2 图片内容为实物部分**

3.2.1训练集

ShiftScaleRotate(interpolation = cv2.INTER\_NEAREST), 随机平移、缩放、旋转图片

OneOf([A.VerticalFlip(p=0.3),HorizontalFlip(p=0.3)]), 水平翻转或垂直翻转

RandomBrightnessContrast(p=0.5), 随机亮度变化

HueSaturationValue(p=0.5), 随机色调、饱和度、值变化

Cutout(2, p=0.5, max\_h\_size = 20, max\_w\_size = 20), 在图像中生成正方形区域

ToGray(p=0.2), 转化为灰度图

Resize(height = 224, width = 224, p=1), 将图像调整为224 \* 224

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

3.2.2测试集

Resize(height = 224, width = 224, p=1), 将图像调整为224 \* 224

Normalize 归一化 将像素值除以255 ，减去每个通道的mean并除以每个通道的std

1. **建模流程、模型介绍**

**4.1图片内容为文字部分**

4.1.1迁移模型

Resnet34 的4-9层

4.1.2下游模型更改

1-3层为

torch.nn.Conv2d(3,64,kernel\_size=(1,1),stride=(1,1),padding=(0,0),bias=False),

torch.nn.BatchNorm2d(64),

torch.nn.ReLU(inplace=True)

Fc 层为

torch.nn.Linear(in\_features=512, out\_features=n\_classes, bias = True)

4.1.3模型总架构

----------------------------------------------------------------

Layer (type) Output Shape Param #

================================================================

Conv2d-1 [-1, 64, 32, 32] 192

BatchNorm2d-2 [-1, 64, 32, 32] 128

ReLU-3 [-1, 64, 32, 32] 0

Conv2d-4 [-1, 64, 32, 32] 36,864

BatchNorm2d-5 [-1, 64, 32, 32] 128

ReLU-6 [-1, 64, 32, 32] 0

Conv2d-7 [-1, 64, 32, 32] 36,864

BatchNorm2d-8 [-1, 64, 32, 32] 128

ReLU-9 [-1, 64, 32, 32] 0

BasicBlock-10 [-1, 64, 32, 32] 0

Conv2d-11 [-1, 64, 32, 32] 36,864

BatchNorm2d-12 [-1, 64, 32, 32] 128

ReLU-13 [-1, 64, 32, 32] 0

Conv2d-14 [-1, 64, 32, 32] 36,864

BatchNorm2d-15 [-1, 64, 32, 32] 128

ReLU-16 [-1, 64, 32, 32] 0

BasicBlock-17 [-1, 64, 32, 32] 0

Conv2d-18 [-1, 64, 32, 32] 36,864

BatchNorm2d-19 [-1, 64, 32, 32] 128

ReLU-20 [-1, 64, 32, 32] 0

Conv2d-21 [-1, 64, 32, 32] 36,864

BatchNorm2d-22 [-1, 64, 32, 32] 128

ReLU-23 [-1, 64, 32, 32] 0

BasicBlock-24 [-1, 64, 32, 32] 0

Conv2d-25 [-1, 128, 16, 16] 73,728

BatchNorm2d-26 [-1, 128, 16, 16] 256

ReLU-27 [-1, 128, 16, 16] 0

Conv2d-28 [-1, 128, 16, 16] 147,456

BatchNorm2d-29 [-1, 128, 16, 16] 256

Conv2d-30 [-1, 128, 16, 16] 8,192

BatchNorm2d-31 [-1, 128, 16, 16] 256

ReLU-32 [-1, 128, 16, 16] 0

BasicBlock-33 [-1, 128, 16, 16] 0

Conv2d-34 [-1, 128, 16, 16] 147,456

BatchNorm2d-35 [-1, 128, 16, 16] 256

ReLU-36 [-1, 128, 16, 16] 0

Conv2d-37 [-1, 128, 16, 16] 147,456

BatchNorm2d-38 [-1, 128, 16, 16] 256

ReLU-39 [-1, 128, 16, 16] 0

BasicBlock-40 [-1, 128, 16, 16] 0

Conv2d-41 [-1, 128, 16, 16] 147,456

BatchNorm2d-42 [-1, 128, 16, 16] 256

ReLU-43 [-1, 128, 16, 16] 0

Conv2d-44 [-1, 128, 16, 16] 147,456

BatchNorm2d-45 [-1, 128, 16, 16] 256

ReLU-46 [-1, 128, 16, 16] 0

BasicBlock-47 [-1, 128, 16, 16] 0

Conv2d-48 [-1, 128, 16, 16] 147,456

BatchNorm2d-49 [-1, 128, 16, 16] 256

ReLU-50 [-1, 128, 16, 16] 0

Conv2d-51 [-1, 128, 16, 16] 147,456

BatchNorm2d-52 [-1, 128, 16, 16] 256

ReLU-53 [-1, 128, 16, 16] 0

BasicBlock-54 [-1, 128, 16, 16] 0

Conv2d-55 [-1, 256, 8, 8] 294,912

BatchNorm2d-56 [-1, 256, 8, 8] 512

ReLU-57 [-1, 256, 8, 8] 0

Conv2d-58 [-1, 256, 8, 8] 589,824

BatchNorm2d-59 [-1, 256, 8, 8] 512

Conv2d-60 [-1, 256, 8, 8] 32,768

BatchNorm2d-61 [-1, 256, 8, 8] 512

ReLU-62 [-1, 256, 8, 8] 0

BasicBlock-63 [-1, 256, 8, 8] 0

Conv2d-64 [-1, 256, 8, 8] 589,824

BatchNorm2d-65 [-1, 256, 8, 8] 512

ReLU-66 [-1, 256, 8, 8] 0

Conv2d-67 [-1, 256, 8, 8] 589,824

BatchNorm2d-68 [-1, 256, 8, 8] 512

ReLU-69 [-1, 256, 8, 8] 0

BasicBlock-70 [-1, 256, 8, 8] 0

Conv2d-71 [-1, 256, 8, 8] 589,824

BatchNorm2d-72 [-1, 256, 8, 8] 512

ReLU-73 [-1, 256, 8, 8] 0

Conv2d-74 [-1, 256, 8, 8] 589,824

BatchNorm2d-75 [-1, 256, 8, 8] 512

ReLU-76 [-1, 256, 8, 8] 0

BasicBlock-77 [-1, 256, 8, 8] 0

Conv2d-78 [-1, 256, 8, 8] 589,824

BatchNorm2d-79 [-1, 256, 8, 8] 512

ReLU-80 [-1, 256, 8, 8] 0

Conv2d-81 [-1, 256, 8, 8] 589,824

BatchNorm2d-82 [-1, 256, 8, 8] 512

ReLU-83 [-1, 256, 8, 8] 0

BasicBlock-84 [-1, 256, 8, 8] 0

Conv2d-85 [-1, 256, 8, 8] 589,824

BatchNorm2d-86 [-1, 256, 8, 8] 512

ReLU-87 [-1, 256, 8, 8] 0

Conv2d-88 [-1, 256, 8, 8] 589,824

BatchNorm2d-89 [-1, 256, 8, 8] 512

ReLU-90 [-1, 256, 8, 8] 0

BasicBlock-91 [-1, 256, 8, 8] 0

Conv2d-92 [-1, 256, 8, 8] 589,824

BatchNorm2d-93 [-1, 256, 8, 8] 512

ReLU-94 [-1, 256, 8, 8] 0

Conv2d-95 [-1, 256, 8, 8] 589,824

BatchNorm2d-96 [-1, 256, 8, 8] 512

ReLU-97 [-1, 256, 8, 8] 0

BasicBlock-98 [-1, 256, 8, 8] 0

Conv2d-99 [-1, 512, 4, 4] 1,179,648

BatchNorm2d-100 [-1, 512, 4, 4] 1,024

ReLU-101 [-1, 512, 4, 4] 0

Conv2d-102 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-103 [-1, 512, 4, 4] 1,024

Conv2d-104 [-1, 512, 4, 4] 131,072

BatchNorm2d-105 [-1, 512, 4, 4] 1,024

ReLU-106 [-1, 512, 4, 4] 0

BasicBlock-107 [-1, 512, 4, 4] 0

Conv2d-108 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-109 [-1, 512, 4, 4] 1,024

ReLU-110 [-1, 512, 4, 4] 0

Conv2d-111 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-112 [-1, 512, 4, 4] 1,024

ReLU-113 [-1, 512, 4, 4] 0

BasicBlock-114 [-1, 512, 4, 4] 0

Conv2d-115 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-116 [-1, 512, 4, 4] 1,024

ReLU-117 [-1, 512, 4, 4] 0

Conv2d-118 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-119 [-1, 512, 4, 4] 1,024

ReLU-120 [-1, 512, 4, 4] 0

BasicBlock-121 [-1, 512, 4, 4] 0

AdaptiveAvgPool2d-122 [-1, 512, 1, 1] 0

Linear-123 [-1, 3755] 1,926,315

================================================================

Total params: 23,201,771

Trainable params: 23,201,771

Non-trainable params: 0

----------------------------------------------------------------

Input size (MB): 0.01

Forward/backward pass size (MB): 26.47

Params size (MB): 88.51

Estimated Total Size (MB): 114.99

----------------------------------------------------------------

**4.2图片内容为实物部分**

4.2.1迁移模型

Resnet34

4.2.2 下游模型

Fc 层为

torch.nn.Linear(in\_features=512, out\_features=n\_classes, bias = True)

4.2.3模型总架构

----------------------------------------------------------------

Layer (type) Output Shape Param #

================================================================

Conv2d-1 [-1, 64, 32, 32] 192

BatchNorm2d-2 [-1, 64, 32, 32] 128

ReLU-3 [-1, 64, 32, 32] 0

Conv2d-4 [-1, 64, 32, 32] 36,864

BatchNorm2d-5 [-1, 64, 32, 32] 128

ReLU-6 [-1, 64, 32, 32] 0

Conv2d-7 [-1, 64, 32, 32] 36,864

BatchNorm2d-8 [-1, 64, 32, 32] 128

ReLU-9 [-1, 64, 32, 32] 0

BasicBlock-10 [-1, 64, 32, 32] 0

Conv2d-11 [-1, 64, 32, 32] 36,864

BatchNorm2d-12 [-1, 64, 32, 32] 128

ReLU-13 [-1, 64, 32, 32] 0

Conv2d-14 [-1, 64, 32, 32] 36,864

BatchNorm2d-15 [-1, 64, 32, 32] 128

ReLU-16 [-1, 64, 32, 32] 0

BasicBlock-17 [-1, 64, 32, 32] 0

Conv2d-18 [-1, 64, 32, 32] 36,864

BatchNorm2d-19 [-1, 64, 32, 32] 128

ReLU-20 [-1, 64, 32, 32] 0

Conv2d-21 [-1, 64, 32, 32] 36,864

BatchNorm2d-22 [-1, 64, 32, 32] 128

ReLU-23 [-1, 64, 32, 32] 0

BasicBlock-24 [-1, 64, 32, 32] 0

Conv2d-25 [-1, 128, 16, 16] 73,728

BatchNorm2d-26 [-1, 128, 16, 16] 256

ReLU-27 [-1, 128, 16, 16] 0

Conv2d-28 [-1, 128, 16, 16] 147,456

BatchNorm2d-29 [-1, 128, 16, 16] 256

Conv2d-30 [-1, 128, 16, 16] 8,192

BatchNorm2d-31 [-1, 128, 16, 16] 256

ReLU-32 [-1, 128, 16, 16] 0

BasicBlock-33 [-1, 128, 16, 16] 0

Conv2d-34 [-1, 128, 16, 16] 147,456

BatchNorm2d-35 [-1, 128, 16, 16] 256

ReLU-36 [-1, 128, 16, 16] 0

Conv2d-37 [-1, 128, 16, 16] 147,456

BatchNorm2d-38 [-1, 128, 16, 16] 256

ReLU-39 [-1, 128, 16, 16] 0

BasicBlock-40 [-1, 128, 16, 16] 0

Conv2d-41 [-1, 128, 16, 16] 147,456

BatchNorm2d-42 [-1, 128, 16, 16] 256

ReLU-43 [-1, 128, 16, 16] 0

Conv2d-44 [-1, 128, 16, 16] 147,456

BatchNorm2d-45 [-1, 128, 16, 16] 256

ReLU-46 [-1, 128, 16, 16] 0

BasicBlock-47 [-1, 128, 16, 16] 0

Conv2d-48 [-1, 128, 16, 16] 147,456

BatchNorm2d-49 [-1, 128, 16, 16] 256

ReLU-50 [-1, 128, 16, 16] 0

Conv2d-51 [-1, 128, 16, 16] 147,456

BatchNorm2d-52 [-1, 128, 16, 16] 256

ReLU-53 [-1, 128, 16, 16] 0

BasicBlock-54 [-1, 128, 16, 16] 0

Conv2d-55 [-1, 256, 8, 8] 294,912

BatchNorm2d-56 [-1, 256, 8, 8] 512

ReLU-57 [-1, 256, 8, 8] 0

Conv2d-58 [-1, 256, 8, 8] 589,824

BatchNorm2d-59 [-1, 256, 8, 8] 512

Conv2d-60 [-1, 256, 8, 8] 32,768

BatchNorm2d-61 [-1, 256, 8, 8] 512

ReLU-62 [-1, 256, 8, 8] 0

BasicBlock-63 [-1, 256, 8, 8] 0

Conv2d-64 [-1, 256, 8, 8] 589,824

BatchNorm2d-65 [-1, 256, 8, 8] 512

ReLU-66 [-1, 256, 8, 8] 0

Conv2d-67 [-1, 256, 8, 8] 589,824

BatchNorm2d-68 [-1, 256, 8, 8] 512

ReLU-69 [-1, 256, 8, 8] 0

BasicBlock-70 [-1, 256, 8, 8] 0

Conv2d-71 [-1, 256, 8, 8] 589,824

BatchNorm2d-72 [-1, 256, 8, 8] 512

ReLU-73 [-1, 256, 8, 8] 0

Conv2d-74 [-1, 256, 8, 8] 589,824

BatchNorm2d-75 [-1, 256, 8, 8] 512

ReLU-76 [-1, 256, 8, 8] 0

BasicBlock-77 [-1, 256, 8, 8] 0

Conv2d-78 [-1, 256, 8, 8] 589,824

BatchNorm2d-79 [-1, 256, 8, 8] 512

ReLU-80 [-1, 256, 8, 8] 0

Conv2d-81 [-1, 256, 8, 8] 589,824

BatchNorm2d-82 [-1, 256, 8, 8] 512

ReLU-83 [-1, 256, 8, 8] 0

BasicBlock-84 [-1, 256, 8, 8] 0

Conv2d-85 [-1, 256, 8, 8] 589,824

BatchNorm2d-86 [-1, 256, 8, 8] 512

ReLU-87 [-1, 256, 8, 8] 0

Conv2d-88 [-1, 256, 8, 8] 589,824

BatchNorm2d-89 [-1, 256, 8, 8] 512

ReLU-90 [-1, 256, 8, 8] 0

BasicBlock-91 [-1, 256, 8, 8] 0

Conv2d-92 [-1, 256, 8, 8] 589,824

BatchNorm2d-93 [-1, 256, 8, 8] 512

ReLU-94 [-1, 256, 8, 8] 0

Conv2d-95 [-1, 256, 8, 8] 589,824

BatchNorm2d-96 [-1, 256, 8, 8] 512

ReLU-97 [-1, 256, 8, 8] 0

BasicBlock-98 [-1, 256, 8, 8] 0

Conv2d-99 [-1, 512, 4, 4] 1,179,648

BatchNorm2d-100 [-1, 512, 4, 4] 1,024

ReLU-101 [-1, 512, 4, 4] 0

Conv2d-102 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-103 [-1, 512, 4, 4] 1,024

Conv2d-104 [-1, 512, 4, 4] 131,072

BatchNorm2d-105 [-1, 512, 4, 4] 1,024

ReLU-106 [-1, 512, 4, 4] 0

BasicBlock-107 [-1, 512, 4, 4] 0

Conv2d-108 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-109 [-1, 512, 4, 4] 1,024

ReLU-110 [-1, 512, 4, 4] 0

Conv2d-111 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-112 [-1, 512, 4, 4] 1,024

ReLU-113 [-1, 512, 4, 4] 0

BasicBlock-114 [-1, 512, 4, 4] 0

Conv2d-115 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-116 [-1, 512, 4, 4] 1,024

ReLU-117 [-1, 512, 4, 4] 0

Conv2d-118 [-1, 512, 4, 4] 2,359,296

BatchNorm2d-119 [-1, 512, 4, 4] 1,024

ReLU-120 [-1, 512, 4, 4] 0

BasicBlock-121 [-1, 512, 4, 4] 0

AdaptiveAvgPool2d-122 [-1, 512, 1, 1] 0

Linear-123 [-1, 3755] 1,926,315

================================================================

Total params: 23,201,771

Trainable params: 23,201,771

Non-trainable params: 0

----------------------------------------------------------------

Input size (MB): 0.01

Forward/backward pass size (MB): 26.47

Params size (MB): 88.51

Estimated Total Size (MB): 114.99

----------------------------------------------------------------

1. **训练方法**

**7.1图片内容为文字部分**

train\_model

Lr\_schedule: ReduceLROnPlateau(

optimizer,

patience=1,

threshold=0.01,

factor = 0.2,

verbose=True,

mode="max")

metric\_func = accuracy

Loss ： cross\_entropy

Optimizer : Adam ：LEARNING\_RATE

最好的权重：checkpoint =

torch.load("../models/gj\_model\_checkpoint/best\_model.pt")

**7.2图片内容为实物部分**

train\_model

Lr\_schedule: ReduceLROnPlateau(

optimizer,

patience=2,

threshold=0.01,

factor = 0.2,

verbose=True,

mode="max")

metric\_func = compute\_accuracy

Loss ： cross\_entropy

Optimizer : Adam ：LEARNING\_RATE

最好的权重：checkpoint =

torch.load("../models/gj\_image\_model\_checkpoint/best\_model.pt")

**Serving**

1. **后端框架介绍**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径gj\_predict

1. **POST ApI 设计**

POST接口调用PyTorch\_REST\_API\_URL + "gj\_predictt"

1. **返回数据示例**

{'prediction': [{'target': '望远镜'}, {'index': 2, 'location': [116, 0, 227, 111]}, {'index': 7, 'location': [0, 232, 111,343]}, {'index': 8, 'location': [116, 232, 227, 343]}], 'success': True}

1. **Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "gj\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/2.png","rb") as f:

base64\_data = base64.b64encode(f.read())

print(gj\_predict\_predict(base64\_data))