第五次实训程序报告

学号: 2310764 姓名: 王亦辉

1 问题重述

基于2020年新冠疫情佩戴口罩的需求，本次实验的核心目标是构建并优化一个深度学习模型。该模型需要能够准确地在图像中检测出人物，并判断他们是否佩戴了口罩。实验要求学习经典的模型结构（如 MTCNN、MobileNet）和训练方法，利用 Python 相关库及深度学习框架实现，最终目标是尽可能提高检测精度，减少人数和口罩佩戴状态的识别误差。

2 设计思想

本实验使用的方法：使用预训练的 MobileNet 模型的参数，在其基础上，使用口罩/无口罩人像图像数据进行微调，使用微调之后的模型执行分类任务。
优化尝试：尝试进行进行 RGB 三通道的归一化，结果发现不如原本使用的归一化方法。对 PyTorch 的学习率调度器中的训练轮次、factor 、patience 参数进行调整，尝试找到能使训练结果最优的参数。

3 代码内容

模型训练代码:

```
import warnings
warnings.filterwarnings('ignore')
import cv2
from PIL import Image
import numpy as np
import copy
import matplotlib.pyplot as plt
from tqdm.auto import tqdm
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision.datasets import ImageFolder
import torchvision.transforms as T
from torch.utils.data import DataLoader
from torch_py.Utils import plot_image
from torch_py.MTCNN.detector import FaceDetector
```

```
from torch_py.MobileNetV1 import MobileNetV1
from torch_py.FaceRec import Recognition
def processing_data(data_path, height=224, width=224, batch_size=32,
                    test_split=0.1):
    """
    数据处理部分
    :param data_path: 数据路径
    :param height:高度
    :param width: 宽度
    :param batch_size: 每次读取图片的数量
    :param test_split: 测试集划分比例
    :return:
    """
    transforms = T.Compose([
        T.Resize((height, width)),
        T.RandomHorizontalFlip(0.1),  # 进行随机水平翻转
        T.RandomVerticalFlip(0.1),  # 进行随机竖直翻转
        T.ToTensor(),  # 转化为张量
        T.Normalize([0], [1]),  # 归一化

    ])

    dataset = ImageFolder(data_path, transform=transforms)
    train_size = int((1-test_split)*len(dataset))
    test_size = len(dataset) - train_size
    train_dataset, test_dataset =
torch.utils.data.random_split(dataset, [train_size, test_size])
    # 创建一个 DataLoader 对象
    train_data_loader = DataLoader(train_dataset,
batch_size=batch_size,shuffle=True)
    valid_data_loader = DataLoader(test_dataset,
batch_size=batch_size,shuffle=True)

    return train_data_loader, valid_data_loader

data_path = './datasets/5f680a696ec9b83bb0037081-momodel/data/image'
train_data_loader, valid_data_loader =
processing_data(data_path=data_path, height=160, width=160,
batch_size=32)

device = torch.device("cuda:0") if torch.cuda.is_available() else
torch.device("cpu")
modify_x, modify_y = torch.ones((32, 3, 160, 160)), torch.ones((32))

```

```python
epochs = 100
model = MobileNetV1(classes=2).to(device)  #加载预训练模型
optimizer = optim.Adam(model.parameters(), lr=1e-3)  # 优化器
print('加载完成...')


# 学习率下降的方式，acc三次不下降就下降学习率继续训练，衰减学习率
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                 'max',
                                                 factor=0.5,
                                                 patience=2)
# 损失函数
criterion = nn.CrossEntropyLoss()


best_loss = 1e9
best_model_weights = copy.deepcopy(model.state_dict())
loss_list = []  # 存储损失函数值

best_val_acc = 0.0
train_loss_list = []
val_loss_list = []
val_acc_list = []

for epoch in range(epochs):
    model.train()
    running_train_loss = 0.0  # 本轮训练 loss 总和

    for batch_idx, (x, y) in tqdm(enumerate(train_data_loader, 1)):
        x = x.to(device)
        y = y.to(device)
        pred_y = model(x)

        loss = criterion(pred_y, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_train_loss += loss.item()

    avg_train_loss = running_train_loss / len(train_data_loader)
    train_loss_list.append(avg_train_loss)

    model.eval()
```

```
    running_val_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for val_x, val_y in valid_data_loader:
            val_x, val_y = val_x.to(device), val_y.to(device)
            val_output = model(val_x)
            val_loss = criterion(val_output, val_y)
            running_val_loss += val_loss.item()

            pred = val_output.argmax(dim=1)
            correct += (pred == val_y).sum().item()
            total += val_y.size(0)

    avg_val_loss = running_val_loss / len(valid_data_loader)
    val_acc = correct / total
    val_loss_list.append(avg_val_loss)
    val_acc_list.append(val_acc)

    # 打印
    print(f"Epoch {epoch + 1}/{epochs} "
          f"|| Train Loss: {avg_train_loss:.4f} "
          f"|| Val Loss: {avg_val_loss:.4f} "
          f"|| Val Acc: {val_acc:.4f}")

    scheduler.step(val_acc)

    # 保存验证集准确率最好的模型
    if val_acc > best_val_acc:
        best_val_acc = val_acc
        best_model_weights = copy.deepcopy(model.state_dict())

# 最后保存最优模型
torch.save(best_model_weights, './results/best_model.pth')
print('Finish Training.')

plt.plot(train_loss_list, label="Train Loss")
plt.plot(val_loss_list, label="Val Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.title("Loss Curve")
plt.show()
```

```
143
144 plt.plot(val_acc_list, label="Val Acc")
145 plt.xlabel("Epoch")
146 plt.ylabel("Accuracy")
147 plt.legend()
148 plt.title("Validation Accuracy Curve")
149 plt.show()
```

进行预测的代码:

```
1  from torch_py.Utils import plot_image
2  from torch_py.MTCNN.detector import FaceDetector
3  from torch_py.MobileNetV1 import MobileNetV1
4  from torch_py.FaceRec import Recognition
5  from torch_py.FaceRec import Recognition
6  from PIL import Image
7  import cv2
8
9  # ------------------------- 请加载您最满意的模型 ----------------------
   -----
10 # 加载模型(请加载你认为的最佳模型)
11 # 加载模型,加载请注意 model_path 是相对路径，与当前文件同级。
12 # 如果你的模型是在 results 文件夹下的 dnn.h5 模型，则 model_path =
   'results/temp.pth'
13 model_path = "results/best_model.pth"
14 # ------------------------------------------------------------------
   -----
15
16 def predict(img):
17     """
18     加载模型和模型预测
19     :param img: cv2.imread 图像
20     :return: 预测的图片中的总人数、其中佩戴口罩的人数
21     """
22     # ------------------------- 实现模型预测部分的代码 -----------------
   ----------
23     # 将 cv2.imread 图像转化为 PIL.Image 图像，用来兼容测试输入的 cv2 读
   取的图像（勿删！！！）
24     # cv2.imread 读取图像的类型是 numpy.ndarray
25     # PIL.Image.open 读取图像的类型是 PIL.JpegImagePlugin.JpegImageFile
26     if isinstance(img, np.ndarray):
27         # 转化为 PIL.JpegImagePlugin.JpegImageFile 类型
28         img = Image.fromarray(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
29
30     recognize = Recognition(model_path)
```

```
31      img, all_num, mask_num = recognize.mask_recognize(img)
32      # ------------------------------------------------------------------
-------
33      return all_num,mask_num
```

4 实验结果

初始实验结果

经过调整参数之后的训练结果，提升了 10 分：

5 总结

也许可以换用更先进的 backbone 来提高模型的基础能力。参数方面，如果可以自动调整并选择的话，也许可以优化得更好。或者也许我们可以多搞一些训练数据，或是更好地进行数据处理，像是数据的归一化（原始是 $\frac{x-0}{1}$ 相当于没有做归一化） 来提升最终模型的能力。

通过本次实验，我对深度学习模型的训练流程和优化方式有了一定了解。