深 圳 大 学 实 验 报 告

课程名称：人工智能课程实训

实验项目名称：实验 1 模型部署实践

学院：计算机与软件学院

专业：软件工程（腾班）

指导教师：王旭

报告人：黄亮铭 学号：2022155028 班级：腾班

实验时间：2024 年 9 月 5 日至 9 月 25 日

实验报告提交时间：2024 年 9 月 22 日

教务处制

实验目的与要求：

目标：

1. 了解模型部署的基本流程和方法
2. 实现将训练的模型部署

基本要求：

1. 基础：基于 streamlit 或 gradio 等开源库实现深度学习模型的部署。
2. 提高：将模型部署在正常生产环境当中（如 Linux 系统下有显卡的场景、手机等 arm 平台、Nvidia jetson 等平台）并且实现模型的稳定运行。

方法、步骤：

参照实验文档给出的 Web UI 模型部署工具一文，使用 Gradio 创建机器学习模型的 Web 页面，进行内容展示。

具体步骤为：

① 定义一个 Web App 脚本，这个脚本的主要作用是调用模型的训练函数，以及简单的设置 Web 页面；

② 在 Github 上下载自己需要部署的模型；

③ 在模型中找到训练函数的位置并将其改写；

④ 启动程序并上传模型需要的文件进行测试。

实验过程及内容：

① 安装相关的依赖（gradio、cuda、pytorch 等）。

I. 在 PowerShell 中输入命令 *pip install gradio* 即可下载，然后再在 PowerShell 中输入命令 *gradio –help* 查看是否安装成功，如果成功则会显示相应的帮助文档（如下图）。

图 1

II. 安装 cuda：首先在命令行中输入 *nvidia-smi* 查看显卡支持的 cuda 版本。

```
C:\Users\黄亮铭>nvidia-smi
Fri Sep 13 17:07:18 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 546.80                 Driver Version: 546.80       CUDA Version: 12.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM  | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=================================+======================+======================|
|   0  NVIDIA GeForce RTX 3060 ...  WDDM  | 00000000:01:00.0  On |                  N/A |
| N/A   51C    P3              23W / 140W |   2424MiB /  6144MiB |     32%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```

图 2

III. 然后在英伟达官网上下载对应版本的 cuda toolkit，然后根据提示安装。安装完成之后重启电脑。

IV. 在 PowerShell 中输入命令 *pip install torch==2.3.1 torchvision==0.18.1 orchaudio==2.3.1 --index-url https://download.pytorch.org/whl/cu121* 即可下载，然后再在 PowerShell 中依次输入命令 *python* | *import torch*。如果没有任何报错提示证明安装成功。

② 在 Github 上下载自己需要部署的模型，这里我使用实验网站提供的暗光增强算法模型（https://github.com/AndersonYong/URetinex-Net）。

I. 在对应网址上鼠标悬浮“Code”，选择下载压缩包。

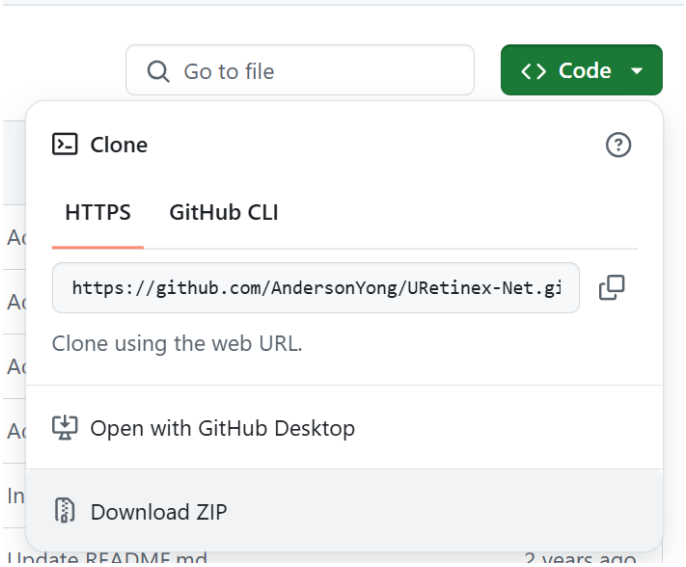

图 3

II. 解压压缩包，并将其移动到相应的路径。

③ 编写 Web App 脚本。

I. 创建 runGradio.py 文件，在其中输入如下内容。

```python
import gradio as gr
import test

# 传入三个参数，必须以关键字参数的形式传入参数
# 第一个参数是训练函数的名字，第二个参数是训练函数的输入类型，
# 第三个参数是训练函数的输出类型
interface=gr.Interface(fn=test.functionForGradio,inputs='image',outputs='image')
# 在这里要注意一下
# 如果指明主机地址是 0.0.0.0，就表示这个应用可以被外部访问
# 如果主机地址是 127.0.0.1，就表示只能被本机应用访问
# gradio 默认的端口是 7860，可是当 7860 被占用时它会使用其他端口
# 如果指明了端口的话，如果端口被占用应用就不会启动，可以起到提示作用
interface.launch(server_name='127.0.0.1',server_port=7860)
```

图 4

④ 改写训练接口。

I. 在模型的 test.py 文件中找到 run 函数，然后将其替换为如下的 runForWeb 函数。修改之处在于在训练前把图片像素压缩，在训练结束后超分辨率。此外，原函数是将结果保存在文件中，这里将结果以数组的形式返回。

```python
def runForWeb(self, image):
    # 对于配置低的服务器可以先对图片下采样训练，再上采样返回
    # 首先对输入的图片进行下采样直到符合最低运行像素限制
    max_pixel_limit=600*600
    pyr_down_times=0
    while True:
        a=len(image)
        b=len(image[0])
        c=a*b
        if(c<=max_pixel_limit):
            break
        pyr_down_times+=1
        image=cv2.pyrDown(image)
    # 对numpy数据预处理
    low_img = self.transform(Image.fromarray(np.uint8(image))).unsqueeze(0)
    # 开始训练
    enhance, p_time = self.forward(input_low_img=low_img)
    # 退训练结果进行上采样，还原原图大小
    result_image = result_for_gradio(enhance)
    for i in range(pyr_down_times):
        result_image = cv2.pyrUp(np.array(result_image))
    # 返回 numpy 类型给 gradio 接口
    return result_image
```

图 5

II. 在 Inference 类外定义 functionForGradio 函数。这个函数的主要作用是初始化模型，并且将模型训练得到的结果返回到 Web 页面进行展示。

```python
def functionForGradio(image):
    parser = argparse.ArgumentParser(description='Configure')
    # specify your data path here!
    parser.add_argument('--img_path', type=str, default="./demo/input/3.png")
    parser.add_argument('--output', type=str, default="./demo/output")
    # ratio are recommended to be 3-5, bigger ratio will lead to over-exposure
    parser.add_argument('--ratio', type=int, default=5)
    # model path
    parser.add_argument('--Decom_model_low_path', type=str, default="./ckpt/init_low.pth")
    parser.add_argument('--unfolding_model_path', type=str, default="./ckpt/unfolding.pth")
    parser.add_argument('--adjust_model_path', type=str, default="./ckpt/L_adjust.pth")
    parser.add_argument('--gpu_id', type=int, default=0)

    opts = parser.parse_args()
    for k, v in vars(opts).items():
        print(k, v)
    os.environ['CUDA_VISIBLE_DEVICES'] = str(opts.gpu_id)
    model = Inference(opts).cuda()
    result_image = model.runForWeb(image)
    return result_image
```

图 6

⑤ 启动程序。在 runGradio.py 文件对应的目录的终端下输入命令 *python runGradio.py* 启动程序。启动程序成功显示如下。

```
PS C:\Users\黄亮铭\Desktop\大学课程\人工智能实训\实验1\module> python runGradio.py
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

图 7

⑥ 上传图片进行测试。

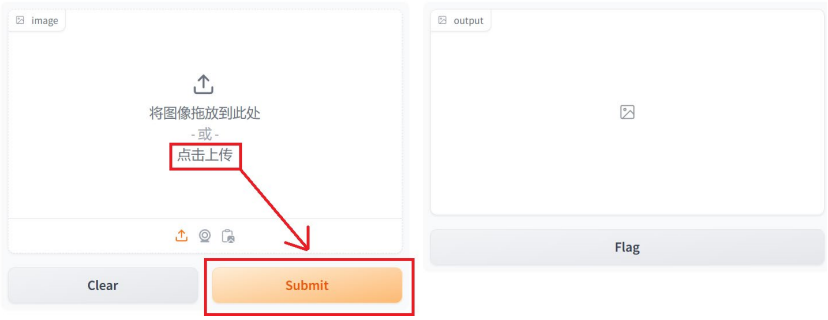

图 8

⑦ 结果展示。

Web 页面显示效果如图所示。

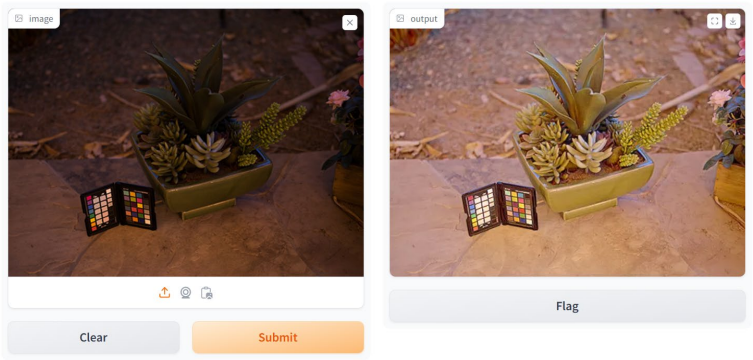

图 9

后台训练输出数据如下图所示。

```
img_path ./demo/input/3.png
output ./demo/output
ratio 5
Decom_model_low_path ./ckpt/init_low.pth
unfolding_model_path ./ckpt/unfolding.pth
adjust_model_path ./ckpt/L_adjust.pth
gpu_id 0
 ===========>  loading pretrained Illumination Adjustment Model from: ./ckpt/L_adjust.pth
Decom(
  (decom): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
  (conv8): Conv2d(64, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
)
Illumination_Alone(
  (conv1): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv2): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv4): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv5): Conv2d(32, 1, kernel_size=(1, 1), stride=(1, 1))
  (leaky_relu_1): LeakyReLU(negative_slope=0.2, inplace=True)
  (leaky_relu_2): LeakyReLU(negative_slope=0.2, inplace=True)
  (leaky_relu_3): LeakyReLU(negative_slope=0.2, inplace=True)
  (leaky_relu_4): LeakyReLU(negative_slope=0.2, inplace=True)
  (relu): ReLU()
)
Adjust_naive(
  (conv1): Conv2d(2, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv2): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv4): Conv2d(32, 1, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (leaky_relu): LeakyReLU(negative_slope=0.2)
  (relu): ReLU()
)
```

图 10

实验结论：

I. 在实验过程中，我成功地使用 Gradio 库将训练好的暗光增强算法模型部署为一个 Web 应用，实现了模型的在线演示和测试。此外，我验证了模型的稳定性，确保了其在生产环境中稳定运行。

II. 在实验中，我遇到了一些挑战，例配置 CUDA 和 PyTorch 环境，以及在模型部署过程中的调试。通过查阅文档和不断尝试，最终解决了问题。

心得体会：

I. 本次实验加深了我对人工智能模型部署的理解。

II. 在实验过程中，我学会了如何使用开源工具 Gradio 来创建 Web 界面，这极大地简化了模型部署的复杂性。

指导教师批阅意见：

成绩评定：

指导教师签字：王旭

2024 年 9 月 18 日

备注：