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：洪子敬 学号：2022155033 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 19 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：

本次实验本人选择部署目标检测模型 Yolo，选择版本为 v5.3.1 版本；由于 cloud studio 上环境较为复杂，部署时较为麻烦，于是部署在 Colab 上，同时利用 streamlit 库实现前端页面的效果展示。实验大致方法步骤如下：

1. 从 github 上 git 克隆 yolo5 模型；
2. 安装 streamlit 库；
3. 安装 yolo5 模型所需的环境要求；
4. 模型的测试和推断；
5. 数据集的下载和解压；
6. 模型训练；
7. 利用 streamlit 库来图形化界面展示。

实验过程及内容：

Yolo 模型本身比较大，而为了不耗费太多资源，我们选用 v5s 较小的模型进行实验。通过创建 run_yolo5 笔记本编写代码进行实验，过程如下：

1. 下载源码

从 github 上 git clone 对应的源码到我们的 colab 环境中即可：

```
[ ] ! git clone https://github.com/ultralytics/yolov5
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 16957, done.
remote: Counting objects: 100% (152/152), done.
remote: Compressing objects: 100% (106/106), done.
remote: Total 16957 (delta 77), reused 98 (delta 46), pack-reused 16805 (from 1)
Receiving objects: 100% (16957/16957), 15.70 MiB | 11.62 MiB/s, done.
Resolving deltas: 100% (11615/11615), done.
```

2. 安装 streamlit 库

直接在笔记本上用 pip 命令安装即可：

```
[ ] ! pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.38.0-py2.py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (from streamlit) (1.4)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (24.1)
```

3. 配置实验环境

克隆的模型中含有 requirements.txt 文件，里面写了运行此模型所需要的所有配置要求，用 pip 命令和-U、-r 选项来安装 yolo5 项目中所需要的所有依赖包，并且确保包为最新版：

```
! pip install -U -r yolov5/requirements.txt

Requirement already satisfied: gitpython>=3.1.30 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 5)) (3.1.43)
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 6)) (3.9.2)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 7)) (1.26.4)
Collecting numpy>=1.23.5 (from -r yolov5/requirements.txt (line 7))
  Using cached numpy-2.1.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 8)) (4.10.0.84)
Requirement already satisfied: pillow>=10.3.0 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 9)) (10.4.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 10)) (6.0.0)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 11)) (6.0.2)
Requirement already satisfied: requests>=2.32.0 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 12)) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 13)) (1.14.1)
Requirement already satisfied: thop>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from -r yolov5/requirements.txt (line 14)) (0.1.1.post2209072238)
```

4. 模型的测试和推断

模型测试的图片是 inference 目录下的图片（当然笔记本上也有写，如果 git 下来后没有此目录要自己创建并手动添加 images 图片），使用的权重数据就是下载的预训练权重参数 yolov5s.pt（可以不用手动下载，运行时如果没有会自己下载），而推断后的输出图片将会存放在新建目录 **runs/detect/exp** 中。

```
%cd yolov5
!python detect.py --source inference/images/ --weights ./yolov5s.pt

/content/yolov5
detect: weights=['./yolov5s.pt'], source=inference/images/, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_t
YOLOv5 🚀 v7.0-366-gf7322921 Python-3.10.12 torch-2.4.1+cu121 CPU

Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 124MB/s]

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
image 1/2 /content/yolov5/inference/images/bus.jpg: 640x480 4 persons, 1 bus, 534.9ms
image 2/2 /content/yolov5/inference/images/zidane.jpg: 384x640 2 persons, 2 ties, 363.3ms
Speed: 2.2ms pre-process, 449.1ms inference, 2.1ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp
```

利用 matplotlib 可视化 images 中检测的图像如下所示：

由大体观察可知，检测的物体还是比较准确的；

5. 数据集的下载和解压

coco128 数据集是 COCO 数据集的前 128 张图片，通常用作小型的教程数据集，这里面的 128 张图片即用作训练也用作验证。下载后要把其解压到与 yolo5 文件夹同一目录（官网上要求的）。具体下载利用 wget 命令配合-P 指定特定目录即可下载：

```
[5] !wget -P /content/ https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip

--2024-09-19 10:37:31-- https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/854f8531-cc3e-47d1-9f20-5
--2024-09-19 10:37:31-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/854f8531-c
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.1
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6983030 (6.7M) [application/octet-stream]
Saving to: '/content/coco128.zip'

coco128.zip          100%[==================>]   6.66M  27.9MB/s    in 0.2s

2024-09-19 10:37:32 (27.9 MB/s) - '/content/coco128.zip' saved [6983030/6983030]
```

配合 unzip 命令和-d 到特定目录即可完成解压：

```
!unzip  /content/coco128.zip  -d  /content/
```

```
Archive:  /content/coco128.zip
   creating: /content/coco128/
  inflating: /content/coco128/LICENSE
   creating: /content/coco128/images/
   creating: /content/coco128/images/train2017/
  inflating: /content/coco128/images/train2017/000000000612.jpg
  inflating: /content/coco128/images/train2017/000000000404.jpg
  inflating: /content/coco128/images/train2017/000000000438.jpg
  inflating: /content/coco128/images/train2017/000000000389.jpg
```

6. 模型训练

上述准备就绪，接下来就要开始最耗时的训练阶段了。训练方式我们采用在预训练权重上训练（默认在 yolo5s.pt 上训练），只训练了 5 个 epoches，效果整体还可以，设置图片大小为 640*640，batch_size 设置为 16。

```
!python  train.py  --img  640  --batch  16  --epochs  5  --data  ./data/coco128.yaml  --cfg  ./models/yolov5s.yaml  --weights  ./yolov5s.pt
```

```
2024-09-19 10:47:49.086860: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attempting to reg
2024-09-19 10:47:49.536325: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempting to re
2024-09-19 10:47:49.669235: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attempting to
train: weights=./yolov5s.pt, cfg=./models/yolov5s.yaml, data=./data/coco128.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=5, batch_size=16,
github: up to date with https://github.com/ultralytics/yolov5
YOLOv5 v7.0-366-gf7322921 Python-3.10.12 torch-2.4.1+cu121 CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.0
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
```

经过较长时间等待，得到最后的训练结果：

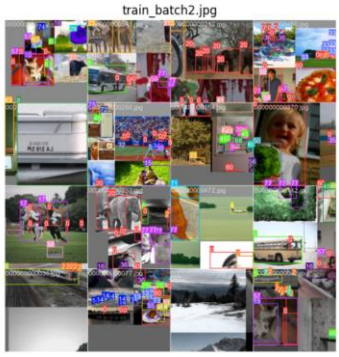
```
5 epochs completed in 0.422 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.9MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.9MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
YOLOv5s summary: 157 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [01:04<00:00, 16.19s/it]
                   all        128        929       0.74      0.681      0.758      0.507
                person        128        254      0.851      0.713       0.81      0.536
               bicycle        128          6      0.906      0.667      0.808      0.452
                   car        128         46      0.699      0.435       0.58      0.237
            motorcycle        128          5       0.93          1      0.995      0.736
              airplane        128          6      0.929          1      0.995       0.71
                   bus        128          7      0.749      0.714      0.761      0.663
```

从输出结果我们可以看到总的以及各种物体的 Images（使用的图像数量）、Instances（检测的目标实例总数）、Precision（准确率）、Recall（召回率，越高代表模型能检测到的目标数量越多）、mAP50（阈值为 0.5 的平均精度）和 mAP50-95（阈值从 0.5 到 0.95 的平均精度）。从整体上看准确率有 74%，召回率有 68%，对老鼠、刀叉、三明治等物体的检测效果较好，当然这也只是 5 个 epoches 的训练，效果整体还行，还可以继续调整参数改进。此外我们还会得到整个过程最好权重 best.pt 和最后的权重 last.py，这两个权重会保存在 runs/exp/weights 目录下，同时此目录下也会保存训练前 3 个 batch 的部分训练数据、预测效果和真实标签。将前 3 个 batch 的数据检测效果用 matplotlib 展示如下：

train_batch1.jpg

train_batch0.jpg

train_batch2.jpg



直观上来看，效果还是比较准确的，没有明显的偏差。

7. 利用 streamlit 库做个简单的可视化网页

前面我们已经把模型调好了也得到了它最好的参数 best.pt，因此我们只需要用 streamlit 编写一个网页，用户通过放入图像，后端带有 best.pt 的模型再去预测得到效果图，接着再将结果展示在前端网页上即可，即做到一个简单的用户交互功能。

```
st.header("图像检测")
st.write("请上传一张图片进行图像检测：")
per_image = st.file_uploader("上传图片", type=['png', 'jpg'], label_visibility='hidden')
col1, col2 = st.columns(2)
with col1:
    if per_image:
        st.image(per_image)
        image = Image.open(per_image)
        image.save("tmp/1/1.jpg")
    else:
        st.image("tmp/2/1.jpg")
    test = st.button("提交图片")

with col2:
    if test and per_image:
        opt.source = "tmp/1/1.jpg"
        run()
        st.image("inference/output/1.jpg")
    elif test:
        opt.source = "tmp/2/1.jpg"
        run()
        st.image("inference/output/1.jpg")
    else:
        st.write("暂无预测结果,请选择图片并点击提交图片")
```

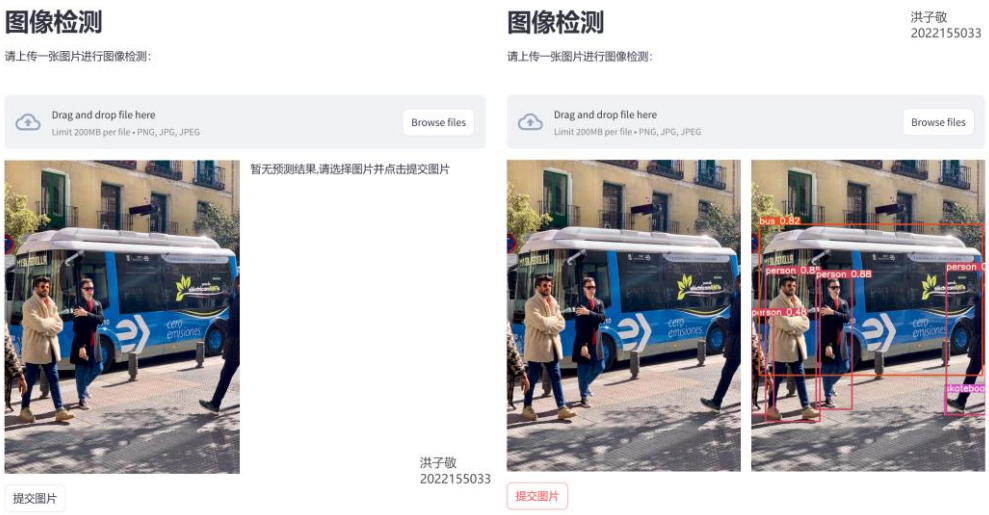
洪子敬
2022155033

此代码主要是编写标题、按钮、文字，特别是分支语句的使用，因为我们想要刚打开网页时会给一张例图，可以直接提交分析，此外我们不想缓冲太多的图片数据到我们的存储上，所以我们采用只存储最近一次分析的结果，其他的通过同名进行覆盖。

由于 colab 上打不开网页（cloud studio 也一样），猜测与内部服务器地址有关，这里直接下载最好权重参数到本地进行运行：

```
(YOLO5) D:\yolo5\yolov5-3.1>streamlit run runStreamlit.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.26.173.139:8501
```

打开网页得到的页面如下左图所示：（这里设置了默认图像，也可自行导入）



直接点击提交结果，得到上面右图结果：我们可以看到，结果还是很正确的；
此外，我们也即可以自行选择图像导入：（点击浏览或者手动拖入均可）

如上图所示，可以看出结果识别效果还是很明显的（为了展示明显，这里不显示全部页面）

至此，本次实验告一段落，当然还可以继续改进，但由于时间有限，这里只给出一些想法：针对老师所讲的，在实际研发中数据标签少，自己标注太花时间，同时也无法保证自己的每次目标识别的结果都是准确，不可能没漏或者没错误。所以我想每次检测结束后可以加入一个**用户反馈 feeedback**，让用户自行选择本次结果是好使坏，如果坏的话，保存本次数据到后台，便于之后的模型的调整分析；这样我们就不用保存每一次结果来判断模型有和不足，而是通过用户的反馈得到结果（当然也要自行筛查，防止用户随便乱选），不用搜图和手动打标签，节省了大量的资源。

实验结论：

本次实验对 Yolo5 模型成功在 colab 上进行部署并训练模型成功，也利用 streamlit 编制了一个网页程序供用户去调用模型，效果也较为不错，本次实验圆满结束。

心得体会：

本次实验第一次进行模型部署，走了很多的弯路，在 cloud studio、paddle 甚至本地都出问题，调了很久本地才能成功运行，最后也是在 colab 这种可以自带环境且不冲突的平台成功部署。通过本次实验，深刻的明白了该怎么样去部署一个模型，最好是创建一个笔记本，用 git clone 去下载模型，通俗的来说就是要学会多用命令，最好不要手动上传，那是最费时费力的方法。

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：