深 圳 大 学 实 验 报 告

课程名称：智能识别系统设计

实验项目名称：基于深度学习的目标识别系统

学院：计算机与软件学院

专业：软件工程（腾班）

指导教师：沈琳琳、文嘉俊

报告人：洪子敬 学号：2022155033 班级：腾班

实验时间：2024 年 12 月 1 日至 2024 年 12 月 28 日

实验报告提交时间：2024 年 12 月 28 日

教务处制

一、实验目的与要求：

1. 实验目的：

1）了解深度学习网络模型（例如 Sphereface 模型或其他相关模型）；

2）了解深度网络特征提取过程（例如 Sphereface 模型或其他相关模型）；

3）把目标检测模块加入工程中，并利用深度网络模型实现对目标特征提取与识别的整个过程。

2. 实验要求：

1）熟悉深度学习工具的编程环境；

2）熟练掌握深度学习网络结构及其设计；

3）熟悉深度网络特征提取与目标识别过程；

二、方法、步骤：

以下问题 2 选 1：

问题 1：

1）利用 OpenCV 深度模型接口调用 Sphereface 模型，并实现对人脸图像的深度特征提取；

2）把 Fast-MTCNN 人脸检测以及基于仿射变换的人脸矫正加入到人脸识别工程中，实现检测，矫正，特征提取与识别的整个流程；

3）人脸数据集中有 10 个类别，在每个类中选取 10 张作业图片作为训练，并利用深度模型进行特征提取，最后计算出识别率。

4）对实验预处理、参数调整、训练以及识别的各个环节作必要分析。

问题 2：

选取合适的深度学习网络，实现对图像或视频中特定目标的识别。

三、实验过程及内容：（其中：提供有简短说明的程序代码。要求：程序运行正确、符合设计要求。）

本实验选择目标检测领域较为常用和热门的 yolov5s 模型实现对个人标注的口算算式数据集进行特征提取和识别。下面是详细的过程：

1. 口算算式数据集介绍

此数据集是事先采集好的，并且进行集体的标注工作得到的；其中我个人也使用 labelme 工具对图片的算式框进行了标注（标注内容为对应的算式的内容），下面是一个标注样例：

图 1. 数据标注过程

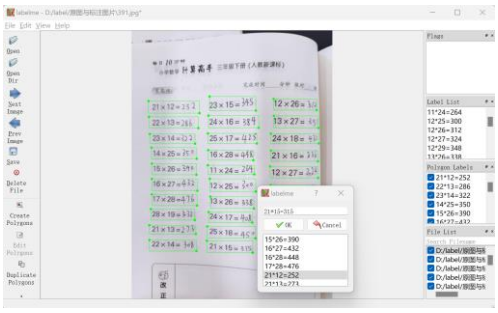

图 2. 数据打标签

标注完成保存为 json 文件后，再使用 Github 开源工具 labelme2yolo 将其格式转换为 yolo 格式（yolo 格式中的 images 是 png 格式，labels 是 txt 格式，并附带有对数据集的 json 说明文档）。

2. Yolov5s 模型介绍

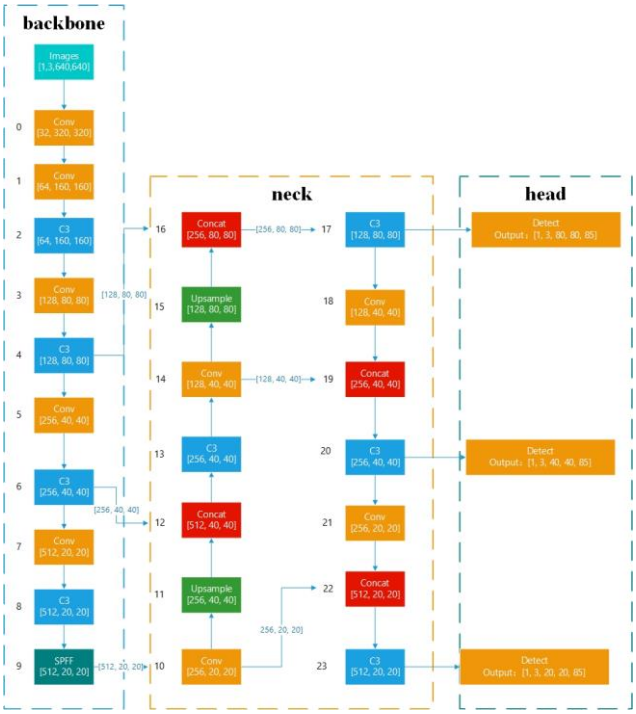

图 3. Yolov5s 模型架构图

如图 3 所示，Yolov5s 的模型架构主要分为三个部分：backbone、neck 和 head，下面对这三部分进行详细的介绍：

a. Backbone

主要功能是从输入图像中**提取特征**，此网络使用了多个卷积层来实现这一点，具体包括：

- 输入层：接收输入图像（通常为 320x320 像素）。
- 卷积层（Conv）：通过多层卷积操作提取图像特征，逐步减少空间维度，增加通道数。
- SPPF（Spatial Pyramid Pooling Fast）：用于增强特征提取能力，通过不同尺度的池化操作捕获多尺度信息。

b. Neck：

主要功能是将 Backbone 提取的特征**进行融合**，以增强不同尺度的特征图，主要包含：

- 上采样层（Upsample）：用来将低分辨率的特征图上采样，以便与高分辨率特征图进行融合。
- Concat：将来自不同层的特征图进行拼接，以提高模型对小目标的检测能力。
- C3（卷积块）：用于进一步处理融合后的特征图，增强特征表示能力。

c. Head:

主要功能是负责生成最终的检测结果，主要包括 Detect 模块，此模块的输出层会生成最终的边界框和类别概率。具体输出包括三个尺度的检测结果，适应不同大小的目标：。

3. 数据集划分

为了后续的训练测试方便，这里先对原始图片进行划分，按照 7：3 的比例划分训练集和测试集（随机数据分布），设置取原始图片的前 500 张图片（即训练集 350 张，测试集 150 张）；在划分好数据集的基础上，将其存放为以下格式便于后续的训练测试：

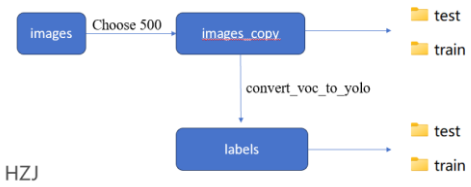

图 4. 数据集格式要求图

4. 数据增强

数据增强(Data Augmentation)是一种常用的数据处理技术，主要用于解决机器学习和深度学习中的数据不足或者数据不平衡问题。数据增强的核心思想是通过对现有数据进行一定程度的变换，生成新的数据样本，从而扩大数据集的规模和样本的多样性，以提高模型的泛化能力。

根据增强方法的不同，数据增强可以分为以下几类：

a) 随机增强：通过随机旋转、翻转、剪裁等方式对现有数据进行处理；

b) 基于模型的增强：通过训练生成模型（如 GAN）来生成高质量数据样本；

c) 基于规则的增强：通过定义一系列规则对现有数据进行处理，生成新数据。

由于做实验的时间较为紧张，本实验采用了较为基础的数据增强方法，即基于 opencv 的方法，包括翻转、加噪声和颜色变换（亮度、对比度变换），并按照 3：3：4 的比例进行图像的随机增强。此外，本实验采用使用训练集中的 20%进行数据增强，图像也是随机挑选的。三个变换函数代码如下：

```python
def augment_flip(image_path, output_path):
    image = Image.open(image_path)
    aug_image = image.transpose(Image.FLIP_LEFT_RIGHT)
    aug_image.save(output_path)

1 usage
def augment_noise(image_path, output_path):
    image = Image.open(image_path)
    np_image = np.array(image)
    noise = np.random.normal( loc: 0,  scale: 25, np_image.shape).astype(np.uint8)
    noisy_image = np.clip(np_image + noise,  a_min: 0,  a_max: 255)
    Image.fromarray(noisy_image).save(output_path)
    # print(f"保存噪声图像到: {output_path}")  # 添加打印语句
```

```python
def augment_color(image_path, output_path):
    image = Image.open(image_path)
    # 随机调整亮度、对比度和饱和度
    enhancer = ImageEnhance.Brightness(image)
    image = enhancer.enhance(random.uniform( a: 0.5,  b: 1.5))
    enhancer = ImageEnhance.Contrast(image)
    image = enhancer.enhance(random.uniform( a: 0.5,  b: 1.5))
    enhancer = ImageEnhance.Color(image)
    image = enhancer.enhance(random.uniform( a: 0.5,  b: 1.5))
    image.save(output_path)
```

图 5. 三个变换函数代码图

具体实现是将选择到的数据增强图片，按照比例随机地进行上述变换，并将变换结果替换原图片，保持训练集大小的一致；此外，由于这些图片均是算式图片，所以标签均是一些算术框的位置，三种变换中只有翻转会改变框的位置，于是将标签中算术框的位置进行对称反转即可，变换代码如下：

```python
filename, yolo_format = convert_voc_to_yolo(xml_file_path, class_names)
output_file_path = os.path.join(output_train_folder, os.path.splitext(xml_file)[0] + '.txt')
with open(output_file_path, 'w') as f:
    for line in yolo_format:
        # 更新标注中的 x_center
        class_id, x_center, y_center, w, h = line.split()
        new_x_center = 1 - float(x_center)  # 翻转后中心点x坐标变化
        f.write(f"{class_id} {new_x_center} {y_center} {w} {h}\n")
```

图 6. 翻转时标签变换代码图

以下是部分图片的变换结果：

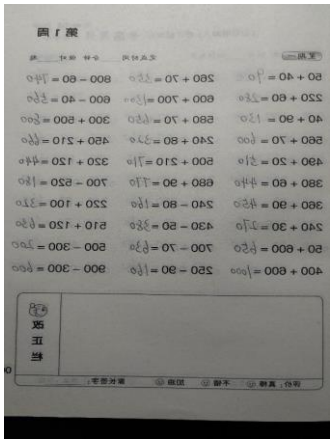

图 7.flip

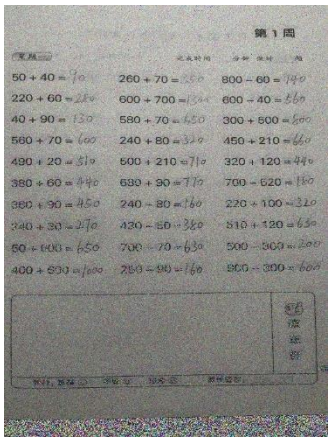

图 8.noise

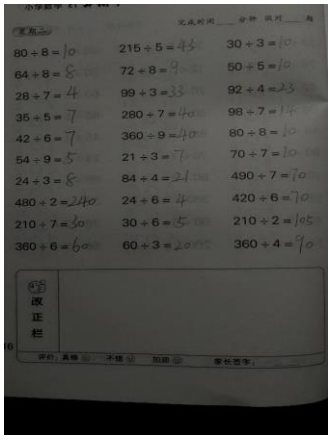

图 9.color

5. 模型训练

虽然 Yolov5s 已经是 Yolo 系列模型中较小的了，但本人电脑没有 GPU 且显存不够，无法进行此模型的训练，所以本实验使用 colab 笔记本进行训练，上面有免费的 GPU 可以用，且可以保存实验记录，刚好符合我们的需求。详细模型部署和训练的步骤如下：

（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.
```

图 10. 模型下载

（2）**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：

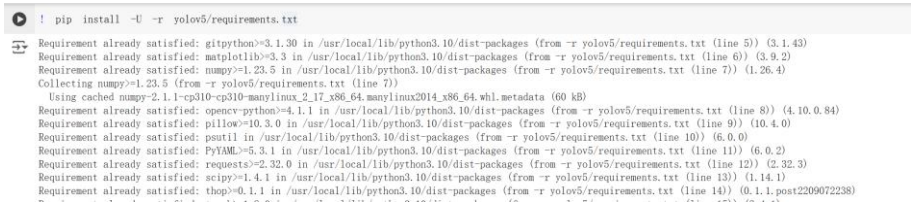

图 11.环境配置

（3）**inference 文件夹的建立**：在 git clone Yolov5s 模型后，由于其没有自带 inference 文件夹，无法使用 detect 函数进行测试，所以我们需要在 yolov5 目录下，新建一个名为“inference”的目录，并在其下再新建一个名为“images”的文件夹，里面可以存放用来测试的图片（格式一般选用“jpg”或者“jpeg”）。

（4）**数据集的存放**：我们需要在 content 目录下，新建一个和 yolov5 同级的文件夹，取名为“datasets”，并将我们想要测试的数据集上传到此目录下（解压缩也要解压到此目录下）。

（5）**配置文件的编写**：我们需要在 data 目录下编写一个新的.yaml 配置文件，对应于我们要训练的数据集。我们实验主要是对算式框的检测，因而只有单一类别是“equation”，标记为 0；剩下的 path、train、val 和 test（可有可无）修改成对应的数据集的路径即可，剩下的均可不要。

（6）**模型训练**：执行下面的命令进入模型训练，训练 5 轮，单次 batch 训练 16 张图片，1 轮大概 22 个 batch，每张图片像素大小为限制为 640，指定预训练权重 yolov5s.pt 和对应训练集的配置文件进行训练：

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

6. 结果分析

增强前后的数据分别用于此模型，每次数据训练在 CPU 上耗时大致在一个小时左右，在 GPU 上耗时大致在六分钟左右。

对于增强前的数据，我们可以得到 5 个 epoches 后的结果如下：

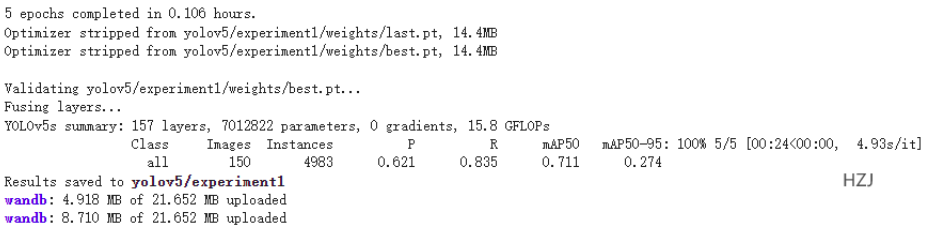

图 12. 数据增强前运行结果图

由上述结果可知，在 150 张测试图像中，识别到 4983 个目标算式框，模型

总的准确率为 62.1%，模型召回率为 83.5%，在 IoU 阈值为 0.5 时的平均精度为 0.711，在 IoU 从 0.5 到 0.95 的平均精度为 0.274，说明此模型在此测试集上能有效识别到目标，但匹配的准确率不高，且在阈值条件较高的情况下表现不佳，模型在检测精度上还有待提升。

对于增强后的数据，我们可以得到 5 个 epoches 后的结果如下：

```
5 epochs completed in 1.080 hours.
Optimizer stripped from yolov5/experiment22/weights/last.pt, 14.4MB
Optimizer stripped from yolov5/experiment22/weights/best.pt, 14.4MB

Validating yolov5/experiment22/weights/best.pt...
Fusing layers...
YOLOv5s summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 5/5 [01:29<00:00, 17.85s/it]
                   all        150       5207      0.713       0.79      0.777      0.261
Results saved to yolov5/experiment22
wandb: 21.502 MB of 21.502 MB uploaded
wandb: 21.502 MB of 21.502 MB uploaded
```

HZJ

图 13. 数据增强后运行结果图

由上述结果可知，在 150 张测试图像中，识别到 5207 个目标算式框，模型总的准确率为 71.3%，模型召回率为 79%，在 IoU 阈值为 0.5 时的平均精度为 0.777，在 IoU 从 0.5 到 0.95 的平均精度为 0.261。

数据增强前后对比可知，模型的精度有所提升，召回率虽然下降，但仍然保持在较高的水平，阈值条件较高时模型表现仍然较差。整体而言，数据增强后的模型相比之前在**精度和整体识别性能有所改善，泛化能力有所提升，整体性能往好的方向发展**，进一步说明了数据增强的作用所在。

7. 结果可视化

为了进一步对实验结果进行展示和分析，使用 W&B 工具进行结果可视化，具体登录网页 https://wandb.ai/进入自己的账号“Project”处，选择进入对应的项目和运行名称即可看到运行过程的可视化结果。

具体实现需要我们在前面进行训练时命令加上“--project”指定项目名称，“--name”指定该项目下的运行名称，比如本次实验设置项目名称为 yolov5s，运行名称为 experiment1、experiment2 等等，如下所示：

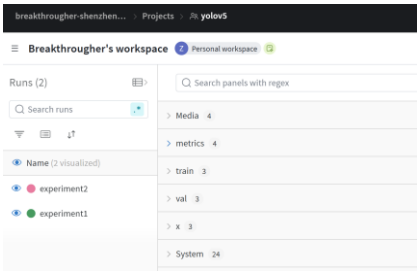

图 14. W&B 网页界面图

使用了 W&B 工具之后，在执行命令、运行结束后会在输出运行的过程信息，包括召回率、精度等的变化，以及过程最好的 epoches 等信息（具体可见提交的.ipynb 文件）；数据增强前后两次实验的输出结果信息如下：

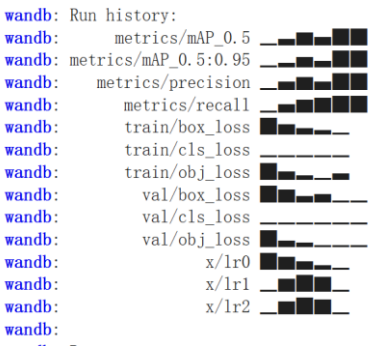

图 15. 数据增强前

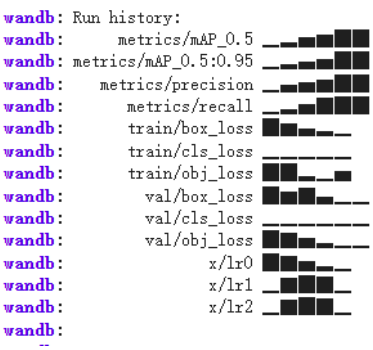

图 16. 数据增强后

除了直接看输出结果以外，我们可以使用 W&B 工具平台到对应的项目运行里面看，该平台上面已经将运行过程数据进行了可视化，结果如下所示：

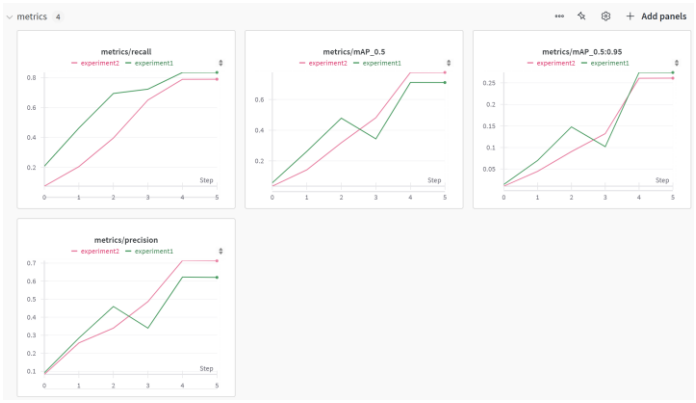

图 17. 召回率、精度、mAP0.5 和 mAP0.5:0.95 的变化图

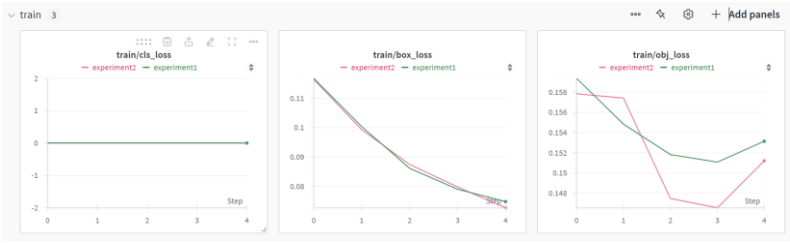

图 18. 训练过程中 cls_loss、box_loss 和 obj_loss 的变化图

图 19. 验证过程中 cls_loss、box_loss 和 obj_loss 的变化图

由图 17 不难发现数据增强后的召回率、精度等各项指标都变化的比较平滑，说明其泛化性能得到了提升，精度和 mAP0.5 的提升侧面佐证了这一点。图 18 显示了训练过程中数据增强后的 loss 会变少，但图 19 显示了验证过程中数据增强后的 loss 又变大了，进一步说明了数据增强带来的一些副作用，增强后的数据不好拟合验证数据集，存在可能使得结果的精度下降的风险，但这是也预料之内

的正常现象，只要精度不下降太多还是可以接受度，舍掉一部分的精度，换取更加鲁棒的模型。

四、实验结论：

本次实验对深度学习模型 Yolov5s 模型进行了了解，包括其特征提取和识别的过程，在此基础上，对该模型进行部署和优化，并使用个人标注的口算算式数据集进行训练集和测试集的划分，同时进行了数据增强，用数据增强前后的数据集成功进行了模型训练，也利用 W&B 工具成功可视化了结果，并对实验结果进行了分析，说明了数据增强的作用。本次实验圆满结束。

五、实验体会：

本次实验从数据集标注和划分，到模型部署和优化，再到模型训练，以及最后的结果分析可视化，形成了一整套项目；通过本实验，学会了如何写配置文件，熟练了在.ipynb 笔记本中运行命令、如何去做简单的数据增强和使用 W&B 工具去观察自己的模型训练结果。通过本次实验，个人对深度学习的模型部署和优化实践有了更加深刻的理解，不再停留于知识层面，提升了个人的动手实践能力，收获满满。

指导教师批阅意见：

成绩评定：

指导教师签字：文嘉俊

2024 年 12 月 31 日

备注：