DETR 测试结果复现报告

仓库简介

这是我本学期（2024秋）数学建模大作业代码仓库，主要复现了 DETR (End-to-End Object Detection with Transformers, ECCV 2020) 这一目标检测领域代表工作。截至目前，该工作已有超过 15000 次引用，影响力极大。

仓库代码基本由 原仓库 fork 得到，我做了如下增改：

1. 将原始 README 文档替换为现在的 README，主要阐述复现过程及复现结果。
2. 整合了一份复现代码 reproduce.py，可以按照下面的操作指令进行复现，且复现精度误差不超过 0.2 个百分点。
3. 对源代码核心部分的注释，主要为 ./model 文件夹下的模型架构代码，包括对源代码的理解。
4. 复现结果 可视化，展示了模型在目标检测和全景分割任务上的效果。可视化的代码与作者提供的 notebook 基本一致，详情参考 DETR's hands on Colab Notebook 和 Panoptic Colab Notebook。

前期准备

1. 下载复现代码至主文件夹。

```
git clone https://github.com/yrrIsYourFather/detr_reproduce.git
```

2. 配置环境。

```
conda create -n detr
conda activate detr
conda install -c pytorch pytorch torchvision
conda install cython scipy
pip install -U
'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
pip install git+https://github.com/cocodataset/panopticapi.git
```

3. 数据集准备：在 COCO 数据集官网 下载 coco2017 和 coco-panoptic 数据集至服务器 data 盘，文件目录结构如下：

```
data2
- coco2017
    - annotations/                    # annotation json files
    - train2017/                      # train images
    - val2017/                        # val images
- coco-panoptic
    - annotations/                  # annotation json files
    - panoptic_train2017/           # train panoptic images
    - panoptic_val2017/             # val panoptic images
```

在主文件夹下建立数据集软链接。

```
ln -s /data2/coco2017 /home/ruiran/detr/coco
ln -s /data2/coco-panoptic /home/ruiran/detr/coco-panoptic
```

4. 作者开源的训练代码可以不做修改直接跑通。但考虑到训练时长和训练资源问题（8 张 V100 并行训练 300 轮次大约需要 6 天），我放弃训练，而是使用作者训练好的模型复现测试结果。

复现测试结果

这一部分我主要在作者源代码 main.py 文件上进行修改，并参考了这一网页，整合得到测试结果复现代码 reproduce.py 。下面我将分别阐释 DETR 在目标检测和全景分割两个任务（分别对应 coco2017 和 coco-panoptic 数据集）的复现方式及结果。

目标检测任务复现

作者共训练了四类 DETR 模型，分别命名为 DETR、DETR-DC5、DETR-R101、DETR-R101-DC5。前两者的 backbone 选用 resnet50，而后两者为 resnet101。“-DC5” 表示 backbone 的 C5 层后添加一个步长为 1 的空洞卷积；这一操作主要用于提高特征分辨率，改善模型对小目标检测的能力。

复现指令

通过在命令行设置不同参数的值实现对不同模型的复现，主要参数意义如下：

- `--batch_size`：每张 GPU 卡上一次测试的图像数，默认为 2。
- `--resume`：作者预训练模型的下载地址。
- `--dataset_file`：数据集名称，默认为 coco，可以设置为 coco-panoptic。
- `--coco_path` / `--coco_panoptic_path`：数据集路径。
- `--dilation`：是否将 backbone C5 层替换为步长为 1 的空洞卷积。
- `--backbone`：骨干网络名称，默认为 resnet50，可以设置为 resnet101。
- `--masks`：是否包含分割头。

1. **DETR**

```
python reproduce.py --no_aux_loss --eval \
    --batch_size 2 \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r50-e632da11.pth \
    --coco_path /home/ruiran/detr/coco
```

2. **DETR-DC5**

```
python reproduce.py --no_aux_loss --eval \
    --batch_size 1 --dilation \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r50-dc5-f0fb7ef5.pth \
    --coco_path /home/ruiran/detr/coco
```

在这一模型的复现过程中，我注意到 `batch_size` 的设置对于复现结果的显著影响。`batch_size=2` 的结果在各项指标上均比 `batch_size=1` 下降了 6~7 个点，例如 `mAP` 由 43.2 降低至 36.0 。我将 DETR-Resnet50 测试阶段的 `batch_size` 同样调整至 1，测试结果却并没有出现类似的大幅波动。我对此感到疑惑，并在作者源代码仓库 issue#217 中找到前人对这一问题的探讨，我了解到：这一问题与模型的训练 `batch_size` 设置紧密相关。DETR-Resnet-DC5 模型训练过程中设置 `batch_size=1`，导致模型从未见过填充（padding），因此在测试集 `batch_size=2`

存在 padding 的情况下性能明显不足；而 DETR-Resnet50 模型训练过程中 `batch_size=4`，不存在这一问题，且模型对不同数量的 padding 具有较好的鲁棒性。

需要说明的是，DC5 模型之所以在训练阶段将 `batch_size` 设置为 1，主要是受到 GPU 内存限制。空洞卷积通过在卷积核元素之间增加空间来扩大感受野，这通常会导致输出特征图的尺寸增加。这也相应地导致了分辨率更高的中间特征图需要在网络的前向和后向传播过程中被存储，从而增加了内存消耗。作者在 issue#129 中解释到，即使是 `batch_size=1` 的 DC5 模型在训练时也需要超过 16GB 的显存，这使得训练阶段更大的 `batch_size` 暂时无法实现。

3. **DETR-R101**

```
python reproduce.py --no_aux_loss --eval \
    --backbone resnet101 \
    --batch_size 2 \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r101-2c7b67e5.pth \
    --coco_path /home/ruiran/detr/coco
```

4. **DETR-R101-DC5**

```
python reproduce.py --no_aux_loss --eval \
    --backbone resnet101 \
    --batch_size 1 --dilation \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r101-dc5-a2e86def.pth \
    --coco_path /home/ruiran/detr/coco
```

复现结果

模型名	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	43.2	63.0	45.9	22.5	47.3	61.0
DETR-R101	43.5	63.8	46.3	21.8	47.9	61.8
DETR-R101-DC5	44.9	64.7	47.7	23.7	49.5	62.3

全景分割任务复现

复现指令

1. **DETR**

```
python reproduce.py --no_aux_loss --eval \
    --batch_size 1 \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r50-panoptic-
00ce5173.pth \
    --masks --dataset_file coco_panoptic \
    --coco_path /home/ruiran/detr/coco \
    --coco_panoptic_path /home/ruiran/detr/coco-panoptic
```

2. **DETR-DC5**

```
python reproduce.py --no_aux_loss --eval \
    --dilation \
    --batch_size 1 \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r50-dc5-panoptic-
da08f1b1.pth \
    --masks --dataset_file coco_panoptic \
    --coco_path /home/ruiran/detr/coco \
    --coco_panoptic_path /home/ruiran/detr/coco-panoptic
```

3. **DETR-R101**

```
python reproduce.py --no_aux_loss --eval \
    --backbone resnet101 \
    --batch_size 1 \
    --resume https://dl.fbaipublicfiles.com/detr/detr-r101-panoptic-
40021d53.pth \
    --masks --dataset_file coco_panoptic \
    --coco_path /home/ruiran/detr/coco \
    --coco_panoptic_path /home/ruiran/detr/coco-panoptic
```

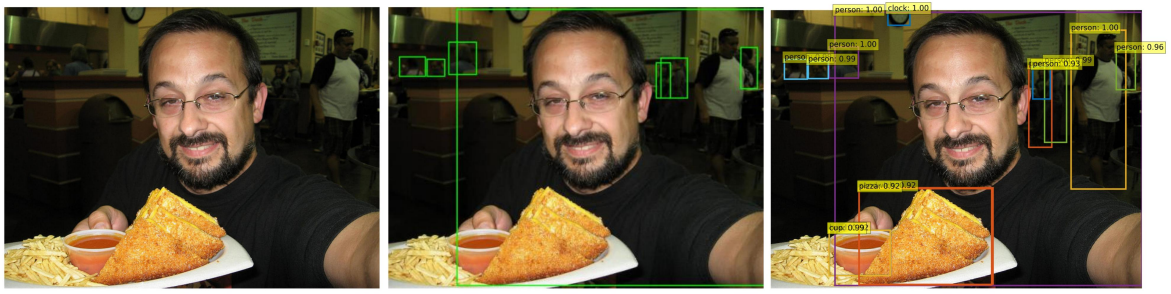
复现结果

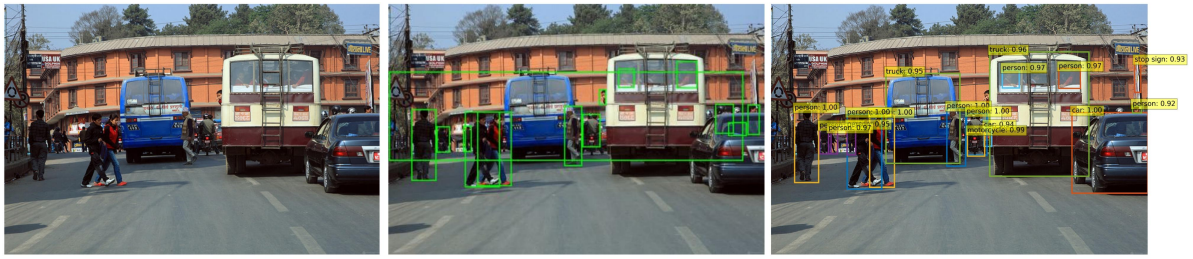
模型名	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
DETR	43.4	79.3	53.8	48.2	79.9	59.5	36.3	78.5	45.3	31.1
DETR-DC5	44.6	79.8	54.9	49.4	80.5	60.6	37.3	78.7	46.4	31.9
DETR-R101	45.1	79.9	55.5	50.6	80.9	61.7	37.0	78.5	46.0	33.0

无论是 DETR 用于目标检测还是全景分割，**复现结果与原文结果误差均不超过 0.2 个百分点，复现效果较为理想，也表明模型在 COCO 系列数据集上确实具有较好的性能。**

除了模型的量化指标复现，我对模型实际效果进行了可视化，如下图所示。其中，前两行为模型在目标检测任务上的效果图，最后一行为全景分割任务。

GT DETR





Mask Panoptic Seg