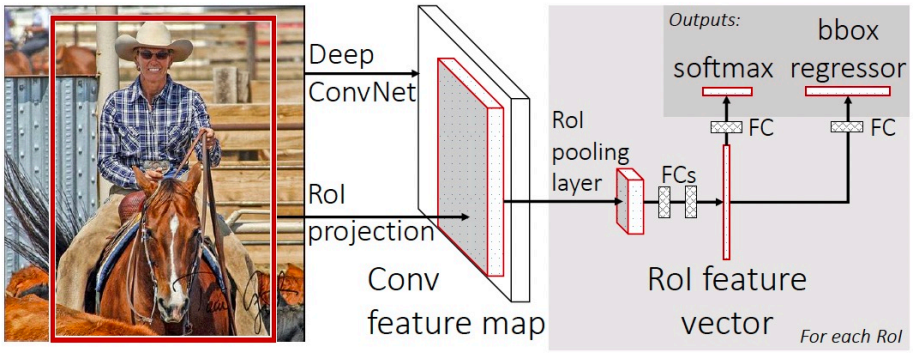
编程作业 2：目标检测

1 作业介绍

目标检测（Object Detection）是计算机视觉的三大基础任务之一，该任务需要同时识别出图像中目标物体的位置（localization）和类别（classification），目标检测被广泛地应用于交通、遥感、视频追踪等多种场景中，为许多视觉任务提供了基础。本次作业将实现一个简单的 Fast R-CNN 目标检测器。

1.1 Fast R-CNN



Fast R-CNN 是一个典型的两阶段检测器。第一阶段为生成候选框（proposal），通常使用传统的计算机视觉方法，如 Selective Search（论文链接）。候选框对应的区域称为 Region-of-Interest（RoI）。第二阶段为识别候选框。Fast R-CNN 使用一个卷积神经网络提取整张图像的特征，随后使用 RoI pooling 将每个 RoI 区域的特征转变为固定大小的特征。RoI 特征通过一系列共享的全连接层，再经过分类和定位预测头，分别得到该候选框的物体类别预测和更精确物体框预测。你可以阅读 Fast R-CNN 的论文来了解更多的细节（论文链接）。

本次作业提供了第一阶段生成好的候选框，需要完成 Fast R-CNN 的第二阶段。

1.2 作业说明

完成本文档中的各个 Task（已用灰色框标记，包括代码实现、文字回答和结果报告）。请将填充好的完整代码和报告文档打包提交，命名方式为“姓名 _ 学号.zip”。

2 作业准备

本次作业使用 VOC2007 目标检测数据集，我们已经为每张图片生成了候选框。本次作业使用 MobileNetV2 作为主干网络，并加载预训练权重作为初始化。

服务器连接：使用以下命令连接远程服务器：（服务器地址、用户名、密码将另外提供）

```
ssh -p username@host_id
```

Conda 环境：使用如下命令激活：
数据文件：储存在路径/data2/hw_data/hw2 下，代码中的默认路径已设置好，无需额外复制。
显卡使用：请严格遵守服务器使用说明中的要求，使用使用分配的指定显卡：

```
CUDA_VISIBLE_DEVICES=gpu_id <your_command>
```

注：如果你希望在本地运行，可以从清华云盘下载本次作业所需的相关文件：云盘链接。数据集压缩包直接解压即可。代码中的数据路径（–data_base_dir 参数）相应修改为下载文件放置的位置。计算量开销：（使用 4090 显卡）本次作业单个实验的正常运行时间不超过 10 分钟。

3 模型构建

3.1 定义模型结构

文件 model.py 定义了 FastRCNN 类，其中在 ___init___ 函数中完成模型各组件的创建。本次作业使用的 Fast R-CNN 包括以下组件：

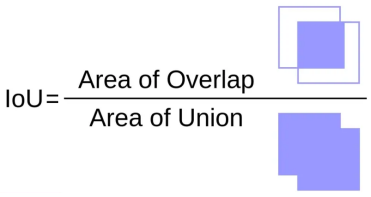
1. 图像特征提取器，由 FeatureExtractor 类定义；
2. 2 层共享全连接网络：linear（输入维度为 in_dim，输出维度为 hidden_dim）-dropout-relu-linear（输入输出维度均为 hidden_dim）；
3. 分类和定位预测头。分类头：linear，输出维度为 20（类别数）+1（背景）。定位头：linear，输出维度为 4。

Task 1
完成 model.py 文件中的 ___init___ 函数。（完成代码即可，不用在报告中写文字说明）

3.2 分配预测目标

在 Fast R-CNN 中，第一阶段的候选框可能包含目标物体，也可能不包含。因此，在第二阶段中，需要模型完成以下事情：（1）确定候选框是否包含物体。（2）如果包含物体，预测物体的类别。（3）如果包含物体，预测更精确的包围框。相应地，在训练时，需要给每一个候选框确定其预测的目标：（a）如果包含物体，则需要预测对应的物体类别和包围框。（b）如果不包含物体，则预测没有物体（背景类）。

首先，为了衡量候选框是否“包含”某一目标物体，我们引入 IoU（Intersection-over-Union）指标。如下图所示，其定义为两个框的重叠面积比共同面积：



Task 2
完成 utils.py 文件中的 compute_iou 函数。(完成代码即可，不用在报告中写文字说明)

接着，utils.py 文件中的 assign_label 函数定义了预测目标的分配。根据候选框和真实框的 IoU，所有候选框将被分为以下三类：

1. 正样本框：与某一目标物体对应，预测目标为该物体的类别，以及候选框与真实框之间的偏移量；
2. 负样本框：对应背景，预测目标为背景类；
3. 其他框：不参与损失计算。为利于模型训练，需要平衡正负样本的比例。通常情况下，对应背景的候选框远多于对应目标物体的候选框，因此从对应背景的候选框中随机选取部分作为负样本，而其他的不计算损失。

注：前景物体的类别 id 为 0-19，具体定义可查看 dataset.py，背景（类）id 为 20。

Task 3
阅读 utils.py 文件中的 assign_label 函数，并简要说明该函数如何判断正负样本框。
阅读 utils.py 文件中的 compute_offsets 函数，简要说明如何计算正样本框到真实框的偏移量。

3.3 完成训练过程

FastRCNN 类中的 forward 函数定义了模型的训练过程，包含以下步骤：

1. 抽取图像特征；
2. RoI pooling，随后对 RoI 特征做空间维度的 average pooling。(可以使用 torchvision 中的 roi_pool 函数)；
3. RoI 特征输入头网络，预测类别分数和偏移量；
4. 对每张图片的候选框完成标签分配（可以使用 utils.py 的 assign_label 函数)；
5. 计算损失。其中，分类损失计算正样本框和负样本框，坐标框回归损失仅计算正样本框。(可以使用 loss.py 的 ClsScoreRegression 函数和 BboxRegression 函数。注意真实的偏移量需要根据候选框和其对应的真实框计算，可以使用 compute_offsets 函数)。

Task 4
完成 model.py 文件中的 forward 函数。(完成代码即可，不用在报告中写文字说明)

3.4 完成推理过程

FastRCNN 类中的 inference 函数定义了模型的推理过程，包含以下步骤：

1. 抽取图像特征；
2. RoI pooling，随后对 RoI 特征做空间维度的 average pooling。(可以使用 torchvision 中的 roi_pool 函数)；
3. RoI 特征输入头网络，预测类别分数和偏移量；
4. 得到每个候选框的预测类别、置信度、以及预测框（使用 utils.py 的 generate_proposal 函数)；

5. 后处理筛选出最终的预测。

对所有候选框，得到类别分数和偏移量预测与训练过程类似。预测类别和置信度由类别分数得到：先计算所有类别的 softmax 概率，随后取除背景类外概率最高的类别为预测类别，对应的概率为置信度。物体预测框为候选框加上偏移量，使用 utils.py 的 generate_proposal 函数得到（该过程是 compute_offsets 的逆过程）。

Task 5
完成 utils.py 的 generate_proposal 函数。（完成代码即可，不用在报告中写文字说明）

其次，在推理时，由于没有真实框，需要使用后处理（post-processing）从所有候选框中筛选出最终的预测。本次作业中需要完成一个简单的后处理过程：

1. 阈值过滤：去掉置信度低于阈值的预测。
2. 极大值抑制（NMS）：对一张图片剩余的预测框（不区分预测类别）进行。（可以使用 torchvision 中的 nms 函数）

Task 6
完成 model.py 的 inference 函数。（完成代码即可，不用在报告中写文字说明）

为了辅助单独检查 inference 过程的正确性，我们提供了一个特定的模型和一组特定的数据点。相关文件储存在路径/data2/hw_data/hw2/inference_check 下（也在清华云盘提供）。

运行如下命令，并将得到的结果与 visualize_refenece 文件夹下的参考进行对照：

```
python check_inference.py
```

注：这一步骤仅为检验代码的辅助参考，不完全代表代码的正确与否。这一步骤非必须，且不纳入作业分数评判。

4 训练和评测

4.1 过拟合实验

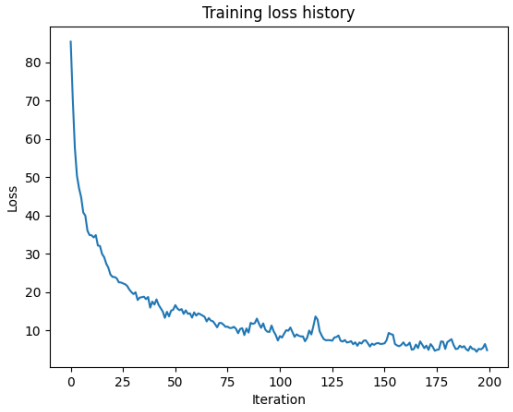
为了快速验证代码的正确性，先在小数据集（10 张图）上观察模型是否能够过拟合。使用以下命令启动实验：

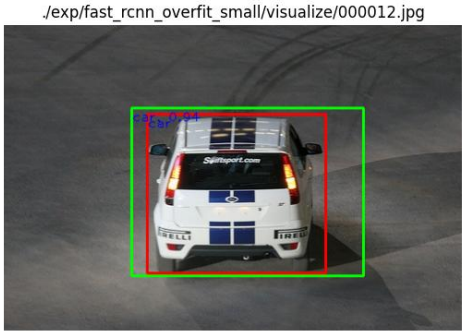
```
python main.py --overfit_small_data --epochs=200
```

训练完成后会自动测试。训练损失曲线、模型 checkpoint 和测试的可视化结果将保存在 output_dir（默认为./exp/fast_rcnn_overfit_small）。

请注意：代码设置了自动保存和读取 checkpoint，即如果 output_dir 路径下存在 checkpoint 文件，代码会自动加载原有模型。如果希望重新训练模型，请改变 output_dir 或者删除原有 checkpoint。

以下是训练曲线和可视化结果的参考，如果你的代码实现正确，将得到类似的结果。





Task 7
完成过拟合实验，在报告中给出训练损失曲线和测试样本可视化。

4.2 最终实验

完成过拟合实验验证后，使用以下命令启动最终的实验：

```
python main.py --epochs=50
```

训练完成后会自动测试。训练损失曲线、模型 checkpoint 和测试输出将保存在 output_dir（默认为./exp/fast_rcnn）。

请注意：代码设置了自动保存和读取 checkpoint，即如果 output_dir 路径下存在 checkpoint 文件，代码会自动加载原有模型。如果希望重新训练模型，请改变 output_dir 或者删除原有 checkpoint。

使用以下命令计算模型的评测结果：（本次作业使用目标检测任务通用的评估指标 mAP）。output_dir 设置为训练的保存路径，如./exp/fast_rcnn。

```
python compute_mAP.py --path=<output_dir>
```

Task 8
完成最终实验，在报告中给出训练损失曲线和评测情况。

受限于算力和时间，本次作业的检测器的 mAP 为 18 左右。通过扩大模型规模、增加训练轮数、使用更先进的检测算法等，现代检测器在 VOC2007 数据集上能够轻松达到 80 以上的 mAP，感兴趣的同学可以自行调研相关文献。