深度学习训练营

案例 7：图像超分辨

1 任务和数据简介

本次案例将使用生成对抗网络来实现 4 倍图像超分辨任务，输入一张低分辨率图像，生成器会生成一张 4 倍超分辨率的图像，如图 1 所示。生成对抗网络选用 SRGAN 结构[1]。本案例训练集使用 DIV2K 数据集[2]，包含有 800 张 2K 左右高分辨率的图像和 800 张对应的低分辨率图像；测试集使用 DIV2K 验证集[2]、Set5、Set14、B100、Urban100 五个数据集，分别包括高分辨率图像和对应的低分辨率图像。训练集和测试集中的所有低分辨图像都是由高分辨率图像下采样得到，下采样方法为使用 Matlab 中的 resize 函数，scale factor 为 0.25，其余为默认参数（双三次插值）。

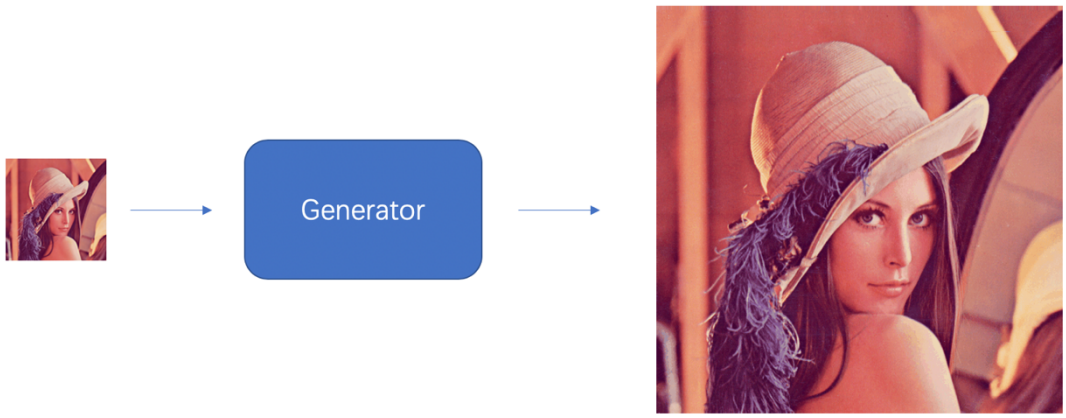

图 1 图像超分辨任务

本案例使用 PSNR 与 SSIM 两个评价指标来衡量生成的高分辨率图像的质量，但指标的高低并不能直接反应图像质量的好坏，因此最终结果评价会加入人工评价，具体见第 4 部分的要求。

2 方法描述

模型结构

案例使用[1]中提出的 SRGAN 结构，生成器和判别器的结构与原论文保持一致，本案例要求自行实现 SRGAN 网络结构。

内容损失函数

本案例涉及到两种内容损失函数，第一种为图像像素空间的 MSE 损失，第二种为图像特征空间的 MSE 损失。

设I^{LR}, I^{HR}分别表示输入的低分辨图像和作为标签的高分辨图像，G(·)表示生成器，D(·)表示判别器。图像像素空间的 MSE 损失表示为

$$L_{MSE}^{SR} = MSE(G(I^{LR}), I^{HR})$$

图像特征空间的 MSE 损失也被称为 VGG 损失，是基于预训练好的 VGG 提取图像特征来计算的。设$\varphi_{i,j}(\cdot)$表示获取 VGG19 网络中第 j 层卷积（包含激活函数）之后，第 i 层最大池化层前的特征图。VGG 损失表示为

$$L_{VGG/i,j}^{SR} = MSE\left(\varphi_{i,j}\left(G(I^{LR})\right), \varphi_{i,j}(I^{HR})\right)$$

对抗损失函数

对抗学习中，生成器生成的高分辨图像要尽可能接近真实的高分辨率图像，能够欺骗判别器识别为真实的高分辨率图像；判别器则要尽可能对生成的高分辨图像和真实的高分辨图像做出区分。因此生成器的对抗损失函数为

$$L_{Gen}^{SR} = -logD\left(G(I^{LR})\right)$$

再加上内容损失L_X^{SR} $(X = MSE\ or\ VGG/i,j)$，生成器的训练损失函数为

$$L_G^{SR} = L_X^{SR} + 10^{-3} L_{Gen}^{SR}$$

判别器是一个二分类器，训练损失函数为交叉熵损失函数。

3 参考程序及使用说明

本案例提供了部分代码供使用，各程序简介如下：

create_data_lists.py：下载好训练集和测试集后，根据提供的数据集地址来生成案例训练测试所需要的 csv 文件。

datasets.py：定义符合 pytorch 标准的 Dataset 类，供读入数据，注意训练阶段每张图片采样了 100 个 patch 来扩充训练集。

imresize.py：用 python 实现了 matlab resize 函数，用于图像下采样。目前 python 第三方包中尚未有能得到与 matlab resize 函数一样结果的函数。

solver.py：定义了一个 epoch 的训练过程。

models.py：定义 SRGAN 模型结构，需要自行实现。

train.ipynb：用于训练的 jupyter 文件，其中超参数需要自行调节，训练过程中可以看到模型损失的变化，每个 epoch 训练后都会进行模型保存。

test.ipynb：加载指定的训练好的模型文件，在 5 个测试集上进行测试，计算并报告各个数据集上的 PSNR 和 SSIM 指标数值。

super_resolution.ipynb：加载指定的训练好的模型文件，针对单个图片进行 4 倍超分辨，并对结果进行可视化。

utils.py：定义了一些可能会用到的函数，如图像数值格式转换等。

环境要求：python 包 pytorch, torchvision, numpy, csv, PIL, matplotlib, easydict, tqdm 等。

使用说明：

1) 下载训练集和测试集[5]，更改 create_data_lists.py 中数据集存放的位置，指定输出文件夹，运行该文件生成案例所需的 csv 文件；

2) 按照 SRGAN 网络结构完成 models.py;

3) 运行 train.ipynb 训练网络, 现在的训练模式为初始化生成器和判别器后, 对生成器和判别器进行交替更新。这样的训练模式只能得到一个表现很差的模型。案例要求自行设计训练模式, 如加入生成器的预训练等[4], 更改 solver.py 和 train.ipynb 训练出一个性能好的模型;

4) 运行 test.ipynb 对训练的模型进行测试, 现在是对 5 个测试集进行 PSNR 和 SSIM 的计算。其中包含了 DIV2K 数据集中的验证集, 这个验证集也可以作为训练时用于调整参数的验证集(如需验证请自行修改 train.ipynb 实现, 不做要求);

5) 模型训练好之后运行 super_resolution.ipynb 生成供人工测评的图片。

4 要求与建议

- **完成 models.py 文件**, 可参考原论文[1];
- 调节 train.ipynb 中的超参数, 使网络结构与原论文保持一致。运行 train.ipynb 使案例可以跑通基础模式的训练;
- **设计生成器和判别器的训练方式**, 可参考[4]中的训练方式, 修改 solver.py 和 train.ipynb 训练出性能更好的模型;
- 运行 test.ipynb 对模型在 5 个测试集上进行测试, 记录 PSNR 与 SSIM 结果;
- 运行 super_resolution.ipynb, 为 **Set5 测试集**中的每一张低分辨图片生成相应的高分辨图片, 保留结果供**人工评价**;
- 完成一个实验报告, 内容包括生成器和判别器的训练方式说明、模型最佳参数和对应的测试集结果、Set5 测试集图片生成结果、自己所做的尝试和改进;
- 提交所有的**代码文件**, 注意 jupyter 文件保留结果, 请不要提交模型文件;

- **禁止任何形式的抄袭**，借鉴开源程序务必加以说明。

5 参考材料

[1] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. https://arxiv.org/abs/1609.04802

[2] https://data.vision.ee.ethz.ch/cvl/DIV2K/

[3] https://zhuanlan.zhihu.com/p/50757421

[4] https://github.com/tensorlayer/srgan

[5] 数据集下载链接

训练集：http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip

http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_LR_bicubic_X4.zip

测试集：http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_HR.zip

http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_LR_bicubic_X4.zip

https://cloud.tsinghua.edu.cn/f/f05ffc6633b44e46947e/?dl=1