

# HW2

赵延顺

2024 年 10 月 27 日

## 1 实验目标

DIP (Pix2Pix) with PyTorch.

## 2 traditional DIP

### 2.1 补充代码

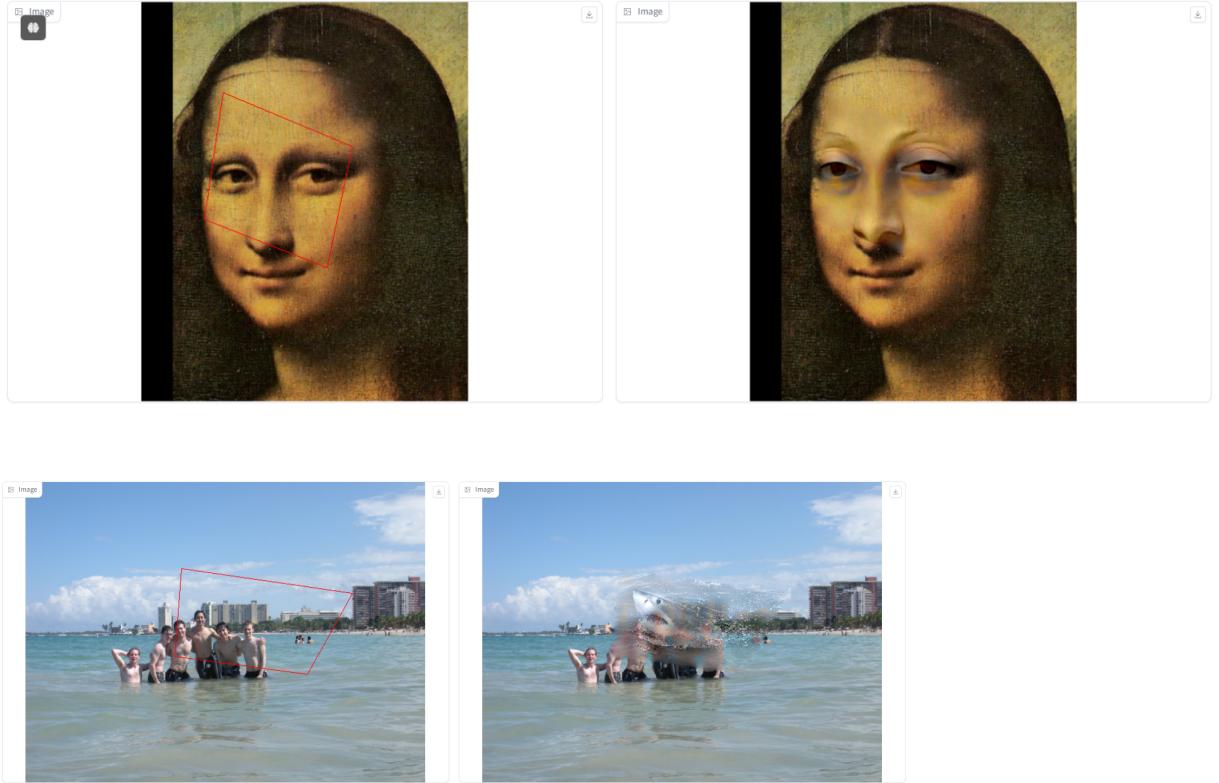
首先是 create mask from points 函数，这个函数的目的是根据给定的多边形顶点生成一个二值 mask，其中 0 表示多边形外部，255 表示多边形内部。

另一个函数是 cal laplacian loss，这个函数的目的是计算前景图像和合成图像（blended image）之间的 Laplacian loss，并且只在给定的掩膜区域内进行计算。

### 2.2 结果

对于提供的三个样例，运行结果为：





### 3 deep learning

#### 3.1 Requirements

首先下载数据集，运行代码：

```
bash download_facades_dataset.sh
```

于是得到了 *facades* 数据集，并将其分成了训练集和测试集。为了进一步测试模型，考虑其他模型，将.sh 文件中的 FILE 变成 *cityscapes*，得到了 *cityscapes* 数据集。

#### 3.2 Training

完善网络的代码，主要是用于编码的卷积层和用于解码的反卷积层，并加入 BatchNorm 层进行正则化，加入非线性激活函数增强模型的拟合能力。

```

    self.conv1 = nn.Sequential(
        nn.Conv2d(3, 8, kernel_size=4, stride=2, padding=1),
        nn.BatchNorm2d(8),
        nn.ReLU(inplace=True)
)

```

图 1: 编码

```

    self.dconv1 = nn.Sequential(
        nn.ConvTranspose2d(64, 32, kernel_size=4, stride=2, padding=1),
        nn.BatchNorm2d(32),
        nn.ReLU(inplace=True)
)

```

图 2: 解码

对于编码器 encoder 和解码器 decoder 都放入 4 个卷积块或反卷积块。  
网络编写完成之后，运行代码：

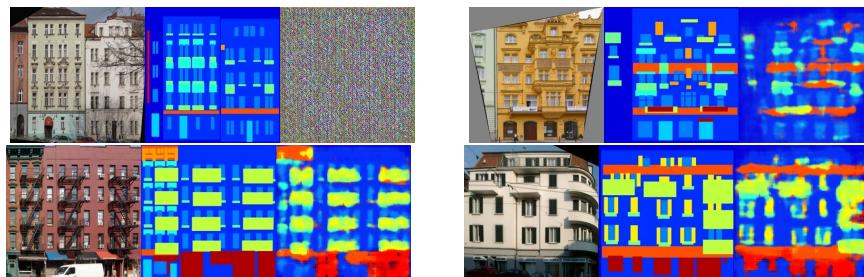
Python train.py

当网络训练完成之后，即可用来评估模型。

### 3.3 Results

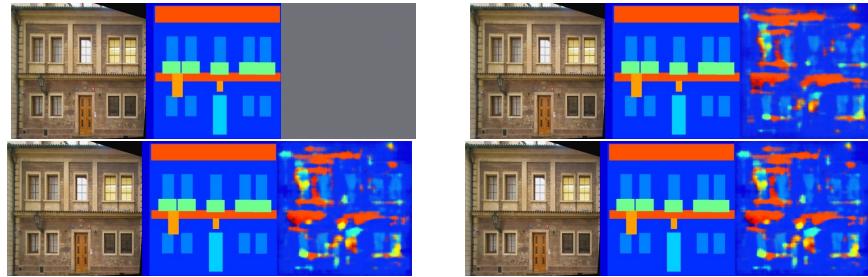
首先是在 facades 数据集进行训练。图片分为 3 部分，分别是输入数据集，目标数据和模型的输出。

在训练集上的效果为：



这 4 张图分别为训练 0 轮、200 轮、500 轮、1000 轮的效果。最终的模型训练的损失为 0.1824。

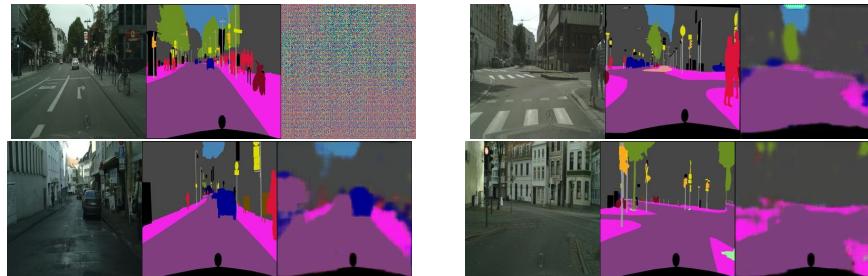
在验证集上的效果为：



这 4 张图分别为训练 0 轮、200 轮、500 轮、1000 轮的效果。最终的模型训练的损失为 0.3823。

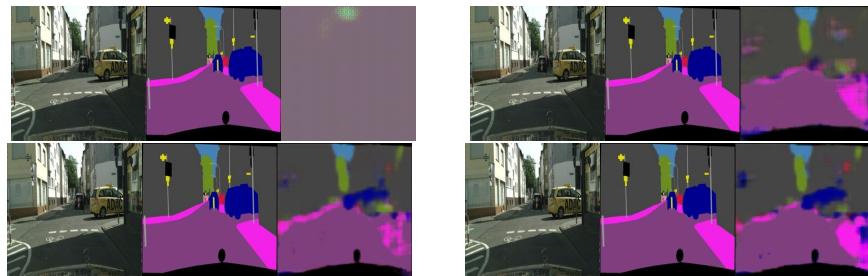
其次是在 cityscapes 数据集进行训练。图片还是分为 3 部分，分别是输入数据集，目标数据和模型的输出。

在训练集上的效果为：



这 4 张图分别为训练 0 轮、50 轮、150 轮、300 轮的效果。最终的模型训练的损失为 0.0956。

在验证集上的效果为：



这 4 张图分别为训练 0 轮、50 轮、150 轮、300 轮的效果。最终的模型训练的损失为 0.1188。

### 3.4 Discussion

从结果来看，在 facades 数据集上，在训练集上的 loss 为 0.1824，在验证集上的 loss 为 0.3823，不仅损失相对较大而且在训练集和验证集的 loss 差距很大，表明过拟合严重；另一方面，在 cityscapes 数据集上，在训练集上的 loss 为 0.0956，在验证集上的 loss 为 0.1188，不仅损失相对较小而且在训练集和验证集的 loss 差距不算太大，表明此时过拟合不太严重。

综上，显然在 cityscapes 数据集上的效果要好于 facades 数据集。这是因为，cityscapes 数据集的数据量要显然高于 facades 数据集，所以模型能学到更有价值、更普遍的信息，从而降低模型的过拟合风险。