哈爾濱工業大學

HARBIN INSTITUTE OF TECHNOLOGY

模式识别与深度学习实验六

DnCNN 去噪模型复现及使用

学　　院：	计算机科学与技术学院	专　　业：	智能信息处理
班　　级：	1903103	学　　号：	1190200527
姓　　名：	王雨桐 石翔宇	实验地点：	
指导老师：	左旺孟	学　　期：	2022 春季学期

2022 年 6 月 8 日

目 录

第 1 章　实验介绍

1.1　选题说明

随着计算机科学和图像处理技术的发展，图像在农业、工业、医学、科研等领域中被广泛使用。然而，图像在成像和传输过程中产生的噪声对图片本身的质量产生了很大的干扰，这可能导致图像的利用价值大打折扣。因此，图像去噪问题成为了底层计算机视觉领域的一个经典问题，旨在设法对图像中的噪声进行抑制和衰减，使得有用的信息得到加强，便会在很大程度上有利于我们对于图像信息的辨别、理解和应用。

假设一张没有噪声的图像（下称原图）$\boldsymbol{x}$，产生了噪声后变为 $\boldsymbol{y}=\boldsymbol{x}+\boldsymbol{n}$。其中，$\boldsymbol{n}$ 是附加在图像上的噪声，这类噪声被称为加性噪声。我们一般假设 $\boldsymbol{n_{ij}} \sim N(0,\sigma^2)$，即作用在每个像素上的噪声值服从均值为 0，方差为 σ^2 的高斯分布，这样的噪声也被称作加性高斯白噪声（AWGN）。其中 σ 在图像去噪问题中又被称作噪声等级，是可以人为设定的参数。图像去噪的目的，就是给定一张含有噪声的图像 $\boldsymbol{y}$，通过机器学习等方法得到尽可能接近原图 $\boldsymbol{x}$ 的图像 $\hat{\boldsymbol{x}}$。

DnCNN 网络是一个十分经典的基于卷积神经网络结构的去噪模型。该模型使用残差学习（Resitual Learning）技术配合批标准化（Batch Normalization）技巧，达到了很好的去噪性能。并且，由于该模型是基于卷积神经网络实现的，因此可以充分利用 GPU 硬件进行加速，这使得该模型在时间性能上也极为可观，对单张图像的去噪时间可以达到亚毫秒级别。而本次实验的内容，就是复现 DnCNN 去噪模型，并使用其对图像进行去噪，并对去噪性能和时间性能进行测试和讨论。

1.2　任务描述及成员分工

本次实验的任务大致分为以下几条：

- 阅读 DnCNN 模型的相关文献，了解 DnCNN 模型的网络架构和设计的技巧点，掌握模型去噪的基本原理；
- 基于 PyTorch 和 CUDA 复现模型的代码，包括训练和测试代码，并对代码进行调试，确保可以对论文结果进行复现；
- 使用训练代码和指定的训练集对模型进行训练，并生成、整理训练的 log 文本；
- 寻找合适的测试集，使用测试代码和测试集对训练好的模型进行测试，记录并整理测试结果；

• 选择合适的评价方法，对测试结果进行评价、讨论，并提出合理的改进意见。

成员王雨桐主要负责第 1、2 条任务，石翔宇主要负责第 3、4 条任务，而第 5 条任务由两人共同完成。

1.3　数据集描述

1.3.1　数据集的选取

在本次实验中，模型的训练集采用了 BSD400 [1] 数据集。该数据集中包含了 400 张 180×180 大小的灰度图,其中部分图片如图 1.1所示。测试集选用了 BSD68 [2] 和 Set12 [3] 数据集。BSD68 由来自 BSD 数据集测试集的 68 张图片组成，大小为 321×481，其中部分图片如图 1.2所示。Set12 由 12 张图片组成，其中 7 张图片大小为 256×256，5 张图片大小为 512×512，其中部分图片如图 1.3所示。

图 1.1　BSD400 中部分照片展示

图 1.2　BSD68 中部分照片展示

图 1.3　Set12 中部分照片展示

1.3.2　数据的预处理

在训练和测试之前，均需要对图片进行归一化处理，具体公式见式 1.1。其中 x_{ij} 表示图像中第 i 行第 j 列位置处的像素值。255 为 8 位像素值所能表示的最大

值，也就是说，该归一化操作将图片的像素值全部转换到 [0, 1] 区间内。

$$x'_{ij} = \frac{x_{ij}}{255.0} \tag{1.1}$$

此外，在对模型进行训练时，我们并不是使用整张图像对模型进行训练，而是首先将图像分成预定大小的 patch，然后为每个 patch 加上高斯噪声，形成训练时的输入，而原来没有噪声的 patch 则是模型的监督数据。根据论文中提出的训练技巧，使用多尺度融合机制和数据增广可以进一步提升模型的训练效果。因此，在对训练数据进行预处理时，还需要将图像进行多种比例的缩放以及多种方式的增广，然后再按照规定的 patch size 裁剪成若干的 patch，作为模型的训练数据。这些预处理工作的具体代码实现见图 1.4。这段预处理程序依次对图像进行 1、0.9、0.8、0.7 比例的缩放，裁剪出 patch 后随机采用翻转、旋转、先旋转后翻转等若干增广方式中的一种对该 patch 进行增广处理。预处理结束后，将所有生成的 patch 作为模型的训练集数据。

```python
def gen_patches(file_name):
    img = cv2.imread(file_name, 0)
    h, w = img.shape
    patches = []
    for s in scales:
        h_scaled, w_scaled = int(h*s), int(w*s)  # 对图像进行多种尺度的缩放
        img_scaled = cv2.resize(img, (h_scaled, w_scaled), interpolation=cv2.INTER_CUBIC)

        for i in range(0, h_scaled-patch_size+1, stride):
            for j in range(0, w_scaled-patch_size+1, stride):
                x = img_scaled[i:i+patch_size, j:j+patch_size]  # 裁剪出规定大小的patch
                for k in range(0, aug_times):
                    x_aug = data_aug(x, mode=np.random.randint(0, 8))  # 随机选取一种数据增广方式
                    patches.append(x_aug)
    return patches
```

```python
def data_aug(img, mode=0):
    '''
    按照随机选取的模型对数据进行增广
    '''
    if mode == 0:  # 不进行任何处理
        return img
    elif mode == 1:  # 上下翻转
        return np.flipud(img)
    elif mode == 2:  # 逆时针旋转90度
        return np.rot90(img)
    elif mode == 3:  # 逆时针旋转90度再上下翻转
        return np.flipud(np.rot90(img))
    elif mode == 4:  #
        return np.rot90(img, k=2)
    elif mode == 5:  # 旋转180度
        return np.flipud(np.rot90(img, k=2))
    elif mode == 6:  # 顺时针旋转90度
        return np.rot90(img, k=3)
    elif mode == 7:  # 顺时针旋转90度再上下翻转
        return np.flipud(np.rot90(img, k=3))
```

图 1.4　部分数据预处理代码实现

第 2 章　实验内容

2.1　方案设计

2.1.1　网络结构

DnCNN 模型的网络结构如图 2.1所示。该网络是一个使用 3×3 大小卷积核，并拥有 17 层卷积层的卷积神经网络。输入层使用 64 个 $3\times3\times c$ 卷积核将输入图像映射为 64 通道的特征图，然后使用 ReLU 作为激活函数。随后是 15 个重复的卷积层，每层卷积都使用 64 个 $3\times3\times64$ 的卷积核进行卷积，使得中间层的通道数始终保持在 64，并且在每次卷积之后都加入 BN（Batch Normalization）层和激活层。输出层使用 1 个 $3\times3\times64$ 的卷积核将 64 通道的特征图最终映射为一个单通道的输出。

假设带有噪声的图像为 $\boldsymbol{x}$，模型期望通过映射 $\mathcal{H}(\cdot)$ 将图片映射成去噪后的图像 $\mathcal{H}(\boldsymbol{x})$。但是由于噪声相对于原本的像素值来说过于微小，这一映射十分接近于恒等映射，这使得模型学习的难度大大增加。因此，该模型选择使用残差学习（Residual Learning）的方法，不直接学习映射 $\mathcal{H}(\cdot)$，转而学习一个残差映射 $\mathcal{F}(\cdot)$，使得 $\mathcal{F}(\boldsymbol{x})=\mathcal{H}(\boldsymbol{x})-\boldsymbol{x}$，即学习去噪后的图像与噪声图之间的残差。由于我们规定施加的噪声为加性高斯白噪声（AWGN），因此该残差刚好可以看作是施加的噪声值的相反数。本质上讲，该模型实际上是在学习图像上的噪声 $-\mathcal{F}(\boldsymbol{x})$，而在进行去噪时，计算 $\mathcal{F}(\boldsymbol{x})+\boldsymbol{x}=\mathcal{H}(\boldsymbol{x})$ 即可得到去噪后的图像 $\mathcal{H}(\boldsymbol{x})$。

该网络中还使用了批标准化（Batch Normalization）以防止样本内部出现的协方差漂移（Internal Covariate Shift）现象，有益于深度较高的网络的训练。况且，由于残差学习的使用，模型实际上是在学习施加在图像上的高斯噪声，而噪声本身就是符合高斯分布的，因此十分有利于批标准化操作的实行。

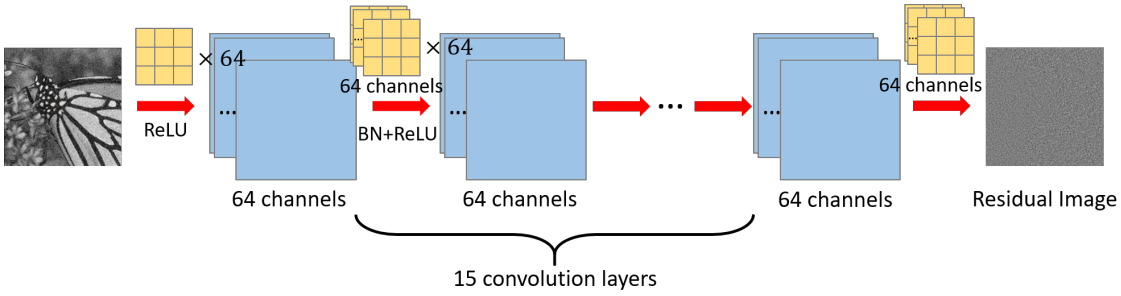

图 2.1　网络结构示意图

2.1.2　损失函数

模型虽然学习的是残差，即施加在图像上的噪声，但最终输出的是将噪声图减去该残差后得到的去噪后图像。我们选择的损失函数应当能够衡量该输出的去噪后图像与原本的 ground-truth 图像的差异。因此，可以选用均方误差损失函数（MSE），具体计算公式见式 2.1。其中 $\boldsymbol{x}$ 表示 ground-truth 图像的像素值组成的矩阵，$\hat{\boldsymbol{x}}$ 表示去噪后图像的像素值组成的矩阵，$\boldsymbol{x}_{i,j}$ 表示 $\boldsymbol{x}$ 的第 i 行第 j 列处的像素值。

$$MSE = \frac{1}{h \times w} \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} \|\boldsymbol{x}(i,j) - \hat{\boldsymbol{x}}(i,j)\|^2 \tag{2.1}$$

2.2　实验过程

2.2.1　DnCNN 结构的实现

本实验首先基于 PyTorch 代码实现了 DnCNN 网络结构，具体代码见图 2.2。由于 DnCNN 网络中有 15 层重复的卷积层，因此在构造网络时使用了一个循环，不断地向层列表中加入对应的模块，最后再将层列表转换为顺序结构。

```
class DnCNN(nn.Module):
    def __init__(self, image_channels=1):
        super(DnCNN, self).__init__()

        layers = []
        layers.append(nn.Conv2d(in_channels=image_channels, out_channels=64, kernel_size=(3, 3), padding=1, bias=True))
        layers.append(nn.ReLU(inplace=True))
        for i in range(15):
            layers.append(nn.Conv2d(in_channels=64, out_channels=64, kernel_size=(3, 3), padding=1, bias=False))
            layers.append(nn.BatchNorm2d(64, eps=0.0001, momentum = 0.95))
            layers.append(nn.ReLU(inplace=True))
        layers.append(nn.Conv2d(in_channels=64, out_channels=image_channels, kernel_size=(3, 3), padding=1, bias=False))
        self.net = nn.Sequential(*layers)
        self._initialize_weights()

    def forward(self, x):
        y = x
        pred_noise = self.net(x)
        denoised_img = y - pred_noise
        return denoised_img
```

图 2.2

2.2.2　训练方案

对模型训练时使用 ADAM 优化器，初始学习率设置为 0.001，经过 30 个 epochs 后学习率减少到 0.0002，共训练 50 个 epochs，训练数据的 batch size 设置为 128。训练过程中，由于我们希望每次施加在训练图像上的噪声都是不同的，因此可以

将对各个 patch 加高斯噪声的工作放在 Dataset 对象的 __getitem__ 方法中，在抽取数据的时候实时添加高斯噪声。具体实现代码如图 2.3所示。

```python
class DenoisingDataset(Dataset):
    def __init__(self, xs, sigma):
        super(DenoisingDataset, self).__init__()
        self.xs = xs
        self.sigma = sigma  # 高斯噪声的方差，也称噪声等级

    def __getitem__(self, index):
        ground = self.xs[index]
        # 为抽取的patch实时添加高斯噪声，除以255是归一化操作
        noise = torch.randn(ground.size()).mul_(self.sigma / 255.0)
        noise_img = ground + noise
        return noise_img, ground

    def __len__(self):
        return self.xs.size(0)
```

图 2.3　为 patch 实时添加噪声的代码实现

训练过程中，对每个 epoch 的 loss 记录 log，并使用 tensorboard 画出 loss 随 epoch 变化的曲线，最终曲线如图 2.4所示。可见，模型在训练到第 35 个 epoch 左右时成功收敛，此时的 MSE 损失约为 1.27。

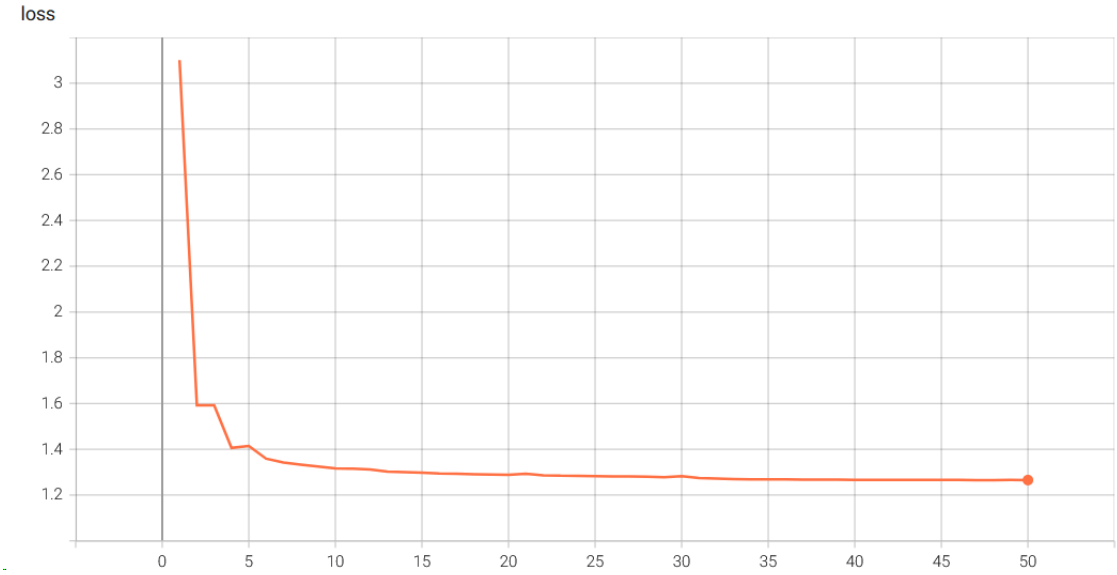

图 2.4　使用 TensorBoard 画出的 epoch-loss 曲线

2.2.3　测试方案

测试时，首先要从训练时保存的文件中读取模型的参数。接下来，对于每一个测试的数据集，我们读取其中的图片。不同于训练时的处理，在测试时我们不再将图像裁剪成多个 patch，而是在整个图片上施加标准差 $\sigma = 25$ 的高斯噪声，再将

整个噪声图输入到模型中，得到降噪后的图片。最后计算得到的图片和原图片的峰值信噪比（PSNR）和结构相似性（SSIM），并记录测试结果。

测试结果见第 3章。

第 3 章　实验结果分析

3.1　测试结果

BSD68 数据集上的部分测试结果如图 3.1所示。

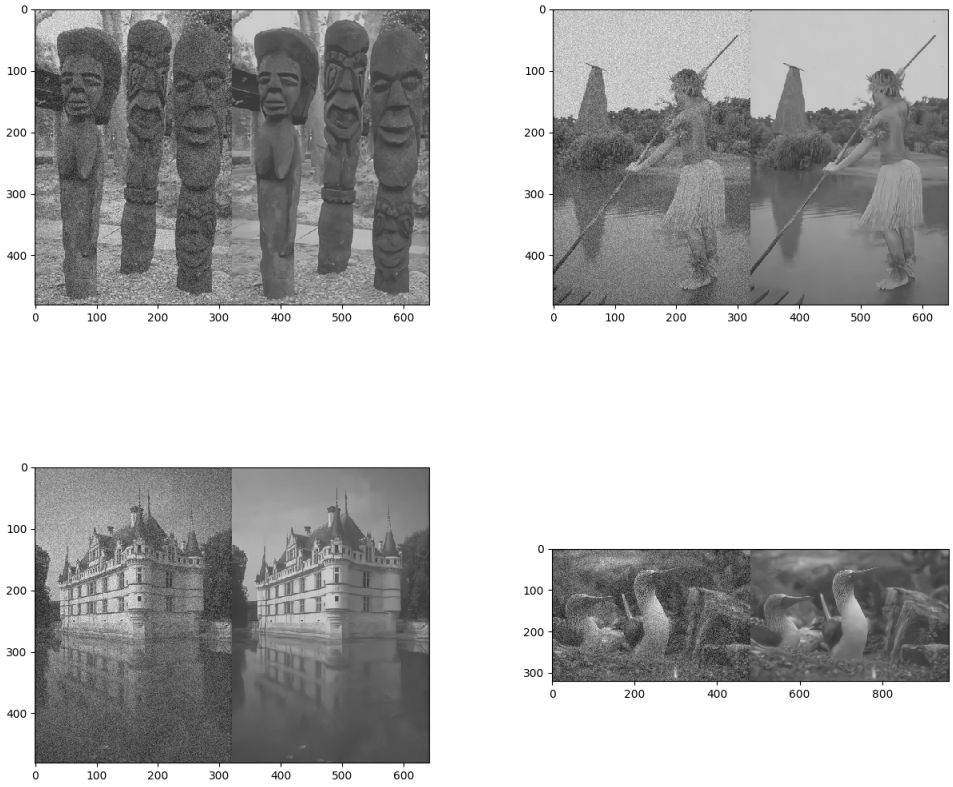

图 3.1　BSD68 中部分测试结果展示

Set12 数据集上的部分测试结果如图 3.2所示。

在两个数据集上去噪结果的平均峰值信噪比（PSNR）和结构相似性（SSIM）如表 3.1所示。

表 3.1　测试结果

	PSNR (dB)	SSIM
BSD68	29.22	0.9011
Set12	30.43	0.9295

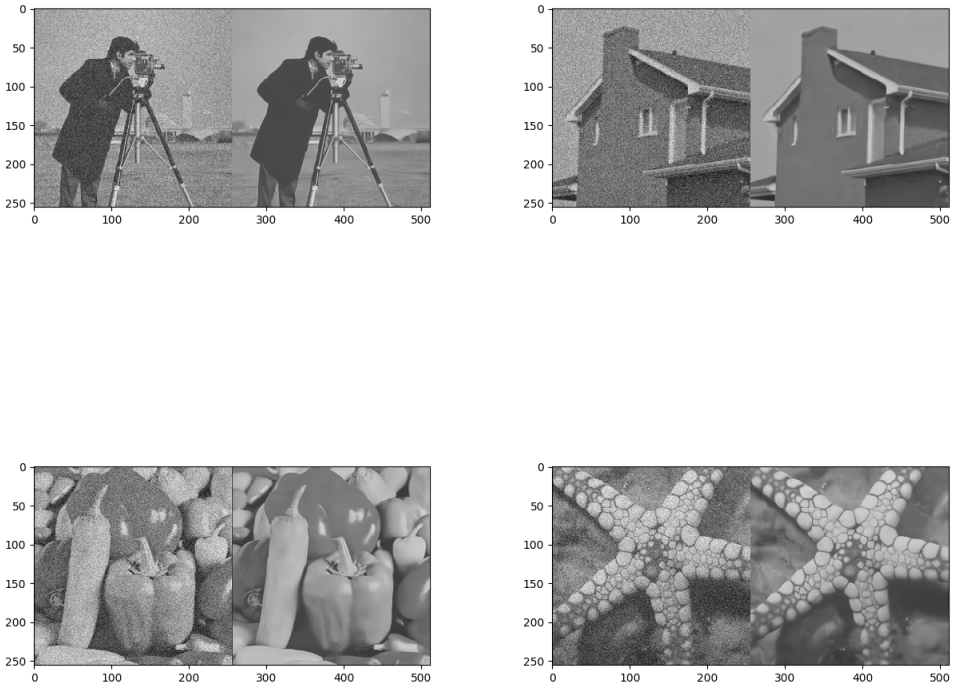

图 3.2　Set12 中部分测试结果展示

3.2　方案评价

从图 3.1和图 3.2中可以看出，无论是大尺寸的照片，还是小尺寸的照片，都能够取得良好的效果。该去噪模型不仅能够将图像中的高斯噪声有效地去除，还能在很大程度上保留图像的细节信息，使得最终能够获得很好的去噪效果。从表 3.1中可以看出，两个数据集上的峰值信噪比只有 30，结构相似性能够达到 0.9 和 0.93 左右，在数值上证明了方法的有效性。

不过，测试集中的某些图片中含有一些重复的图案或规则性的结构，例如图 1.3第二幅图像中的房子表面有大量规则的砖块纹理。类似这样的图像中，具有相似图案的两块图像之间可以相互提供去噪的参考信息，在去噪领域我们称这种信息为非局部（nonlocal）信息。而我们实现的 DnCNN 网络是一个典型的卷积神经网络，去噪时所使用的参考信息大致可认为是感受野中的所有信息，这是在图像中一个连续范围内的局部（local）信息。如果我们能够在该网络结构的基础之上，加入一些与当前去噪区域相似的其他区域当中包含的非局部信息，则对于上述这类有较多重复图案的图像的去噪效果可能会有所提升。

参考文献

[1] Y. Chen and T. Pock, “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1256–1272, 2016.

[2] S. Roth and M. J. Black, “Fields of experts: A framework for learning image priors,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 2. IEEE, 2005, pp. 860–867.

[3] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,” *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.