计算机视觉与应用实践实验报告（三）

目录

一．实验内容

- 实现基于逐像素损失的图像超分辨率算法 SRCNN 在 Set5 数据集上的测试，得到超分辨率图像，并进行分析；
- 实现基于 GAN 的图像超分辨率算法 SRGAN 在 Set5 数据集上的测试，得到超分辨率图像，并进行分析；
- 对比两种类型的图像超分辨率方法在训练过程和生成图像质量上的不同。

二．数据集与评估准则

2.1 数据集

Set5 是常用的图像超分辨率测试数据集，主要用于评估和比较不同图像超分辨率算法的性能。如图 2.1 所示，它包含 5 个测试图像，分别是“baby”、“bird”、“head”、“woman”和“butterfly”。

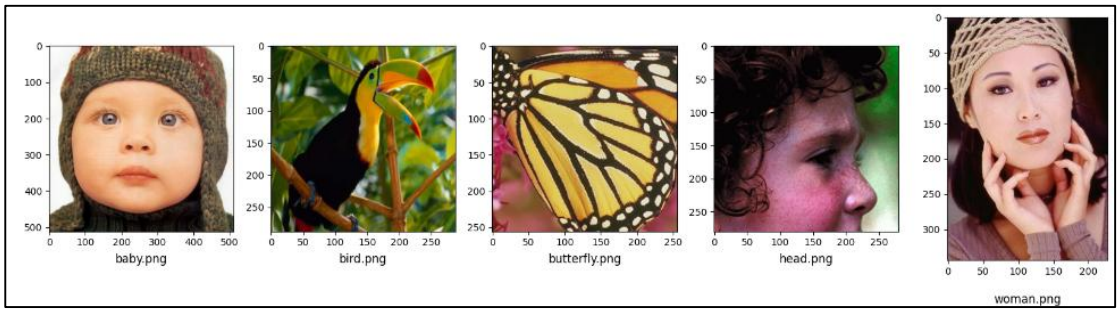

图 2. 1 Set5 数据集

由于 Set5 数据集没有提供高分辨率图像，所以使用 Bicubic 插值法对数据集图像进行下采样，下采样 x4 的对应图像如图 2.2 所示。原图作为 Ground Truth

（真实的高分辨率图像），下采样图像作为输入，对其应用图像超分辨率算法得到超分辨率图像，与原图进行比较。

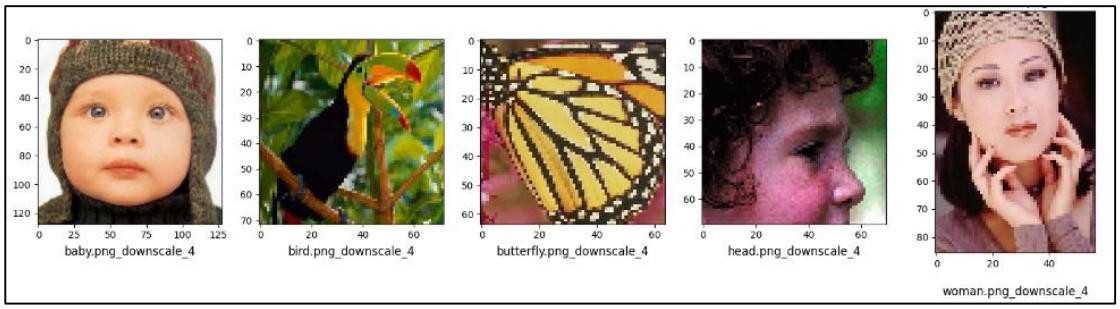

图 2. 2 Set5 下采样

2.2 评估准则

实验中使用峰值信噪比和结构相似性指数作为评估准测：

- **峰值信噪比(Peak Signal to Noise Ratio, PSNR)**

PSNR 是一种评价图像质量的度量标准。因为 PSNR 值具有局限性，所以它只是衡量最大值信号和背景噪音之间的图像质量参考值。PSNR 的单位为 dB，其值越大，图像失真越少。一般来说，PSNR 高于 40dB 说明图像质量几乎与原图一样好；在 30-40dB 之间通常表示图像质量的失真损失在可接受范围内；在 20-30dB 之间说明图像质量比较差；PSNR 低于 20dB 说明图像失真严重。（越高越好）。

- **结构相似性指数（structural similarity index, SSIM）**

SSIM 是一种用于量化两幅图像间的结构相似性的指标。与 L2 损失函数不同，SSIM 仿照人类的视觉系统（Human Visual System, HVS）实现了结构相似性的有关理论，对图像的局部结构变化的感知敏感。SSIM 从亮度、对比度以及结构量化图像的属性，用均值估计亮度，方差估计对比度，协方差估计结构相似程度。SSIM 值的范围为 0 至 1，越大代表图像越相似。如果两张图片完全一样时，SSIM 值为 1。（越高越好）。

三．SRCNN

3.1 算法介绍

SRCNN（Super-Resolution Convolutional Neural Network）是一种端到端的用于图像超分辨率（Image Super-Resolution, ISR）的深度学习算法。其核心思想是利用深度卷积神经网络（CNN）来学习从低分辨率图像中恢复出高分辨率图像

的映射。采用三层卷积神经网络，分别将低像素图像中提取到的特征表示成更高维度的向量然后进行非线性投影，每个投影向量就是高像素图像的分块表示，最后在重构过程中将所有的分块表示聚集在一起构成了高像素的重构恢复图像。算法主要由以下 3 个部分组成：

1. 图像的提取和表示：首先对低分辨率图像 Y 进行特征的提取和表示，通过一系列的卷积操作就可以获得低分辨图像 Y 的特征图，这些特征图表示了低分辨图像 Y，第一层的操作可以表示为 F_1：

$$F_1(Y) = \max(0, W_1 * Y + B_1) \tag{3.1}$$

式中：W_1和B_1分别表示特征提取的卷积参数和偏置量。W_1包含n_1个卷积核，每个卷积核的大小为$c * f_1 * f_1$，其大小为$c * f_1 * f_1 * n_1$，其中 c 为输入图像的通道数。激活函数选择 ReLU 函数，使模型较快完成收敛。

2. 非线性映射：经过第 1 层的运算之后，对每一个图像块都提取了一个n_1维的特征图，第 2 步操作就是将一个n_1维的特征图再映射成为一个n_2维的特征图，相当于通过n_2个$f_2 * f_2$的卷积核卷积进行映射，然后得到一个新的n_2维特征图。第 2 层的操作可以表示为 F_2：

$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2) \tag{3.2}$$

式中：W_2表示非线性映射的卷积参数，W_2包含n_2个卷积核，每个卷积核的大小为$n_1 * f_2 * f_2$，其大小为$n_1 * f_2 * f_2 * n_2$，而B_2是一个n_2维的向量，同样表示偏置。激活函数选择 ReLU 函数。

3. 图像重建：经过第 2 层的非线性映射层之后就已经完成了低分辨率特征到高分辨率特征的映射过程，此时只需要组合高分辨率特征使得得到的目标图像满足 HR 图像的要求就可以了。在图像重建过程中，一般会对第 2 步操作所得到的图像块进行滤波，从而得到目标高分辨率图像。图像的滤波可以通过对这些图像块做均值处理来实现，再采用均值处理后的特征图形成最后的 HR 图像。第 3 层的操作可以表示为 F_3：

$$F_3(Y) = W_3 * F_2(Y) + B_3 \tag{3.3}$$

式中：W_3表示图像重建的卷积参数，W_3包含c个卷积核，每个卷积核的大小为$n_2 * f_3 * f_3$，其大小为$n_2 * f_3 * f_3 * c$，而B_3是一个c维的向量，表示偏置。不使用激活函数。

SRCNN 算法框图如图 3.1 所示。实验中设置 SRCNN 的网络参数为：f_1= 9, f_2= 5, f_3= 5, n_1= 64, n_2= 32。

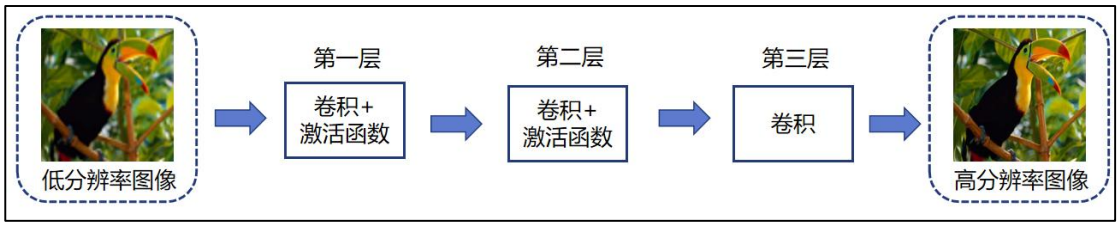

图 3. 1 SRCNN 算法框图

损失函数使用最小化均方误差(MSE)作为优化目标进行模型训练，计算方式如下：

$$L(\theta) = min\frac{1}{n}\sum_{i=1}^{n}\left\|F(Y_L^i;\theta) - X^i\right\|_2^2 \tag{3.4}$$

式中：Y_L^i和X^i分别表示对应的原始低分辨率图像和高分辨率图像，i表示不同的通道，$\theta = \{W_i, B_i\}$表示网络的卷积层数，$F(Y_L^i;\theta)$表示网络的输出结果，n表示训练样本，使用随机梯度下降法来逼近最优解。目的就是使得计算输出的高分辨率图像与原高分辨率图像的像素值差异最小。

3.2 关键实现代码

这一部分给出 SRCNN 在 Set5 数据集上测试的关键实现代码。其中包括模型的搭建与权重加载，数据预处理与测试结果生成。

- 模型搭建：SRCNN 的模型实现代码如图 3.2 所示，实验中设置网络参数为：f_1= 9, f_2= 5, f_3= 5, n_1= 64, n_2= 32，模型只对颜色空间 YCbCr 的 Y 通道进行处理，所以设置 in_channel = out_channel = 1。网络结构图如图 3.3 所示（假设输入图像为 1x64x64），三层卷积层通过设置 padding 值保证图像的分辨率保持不变，其中只对前两层卷积应用激活函数。

```
# f1 = 9, f2 = 5, f3 = 5, n1 = 64, n2 = 32
# 输入是y通道 in_channel = out_channel = 1
class SRCNN(nn.Module):
    def __init__(self):
        super(SRCNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=64, kernel_size=9, padding=4)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=32, kernel_size=5, padding=2)
        self.conv3 = nn.Conv2d(in_channels=32, out_channels=1, kernel_size=5, padding=2)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.conv3(x)
        return x
```

图 3. 2 SRCNN 模型实现代码

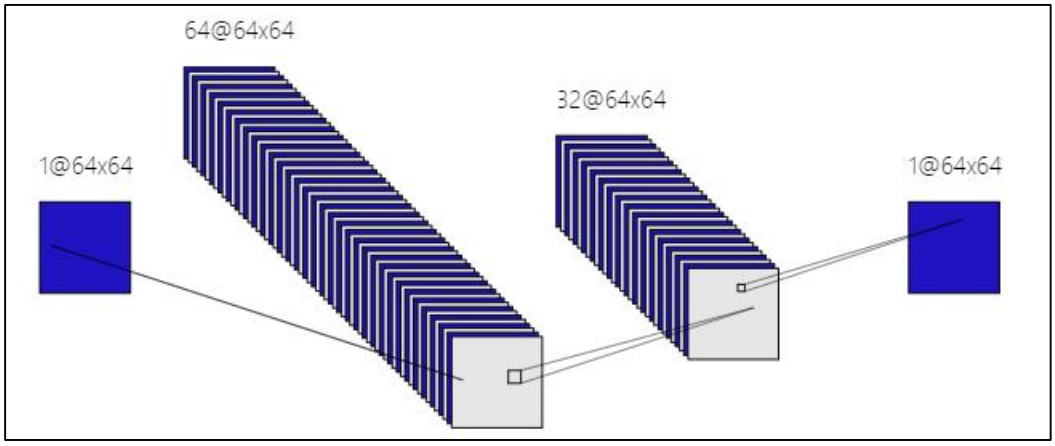

图 3. 3 SRCNN 模型结构

- 权重加载：由于本实验是对 Set5 数据集进行测试，所以如图 3.4 所示，加载训练好的模型权重进行测试，无需从零训练。实验使用 GPU 加快模型计算，读取训练好的.pth 权重文件，并设置模型为测试模式，减少梯度的计算，节省计算开销。

```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
model_weights_path = './checkpoints/srcnn_x4.pth'
model = SRCNN().to(device)
checkpoint = torch.load(model_weights_path, map_location=lambda storage, loc: storage)
# Load the SRCNN weights
model.load_state_dict(checkpoint)
model.eval()
```

图 3. 4 模型权重加载

- 数据预处理：使用 Set5 数据集的下采样图像作为模型输入，由于模型输出的图像大小与输入图像大小一致，所以在输入模型之前，先将图像上采样恢复到与原图大小一致（使用 bicubic 方式进行图像上采样，倍数为 4），再输入模型进行计算。模型只对图像的 y 通道处理，将图像从 RGB 通道转换到 YCbCr 通道，提取 Y 通道归一化，转换成模型要求输入的 Tensor 格式。具体实现代码如图 3.5 所示。

```
image = pil_image.open(lr_image_path).convert('RGB')
# get bicubiced images
image = image.resize((image.width * scale_factor, image.height * scale_factor), resample=pil_image.BICUBIC)
# 对y通道做处理
ycbcr = convert_rgb_to_ycbcr(image)
y = ycbcr[..., 0]
y /= 255.
y = torch.from_numpy(y).to(device)
y = y.unsqueeze(0).unsqueeze(0)
```

图 3. 5 数据预处理

- 生成结果：将处理好的图像数据输入模型，得到输出结果，对其进行通道合并，转换颜色空间，保存结果。实现代码如图 3.6 所示。

```
with torch.no_grad():
    preds = model(y).clamp(0.0, 1.0)

preds = preds.mul(255.0).cpu().numpy().squeeze(0).squeeze(0)
output = np.array([preds, ycbcr[..., 1], ycbcr[..., 2]]).transpose([1, 2, 0])
output = np.clip(convert_ycbcr_to_rgb(output), 0.0, 255.0).astype(np.uint8)
output = pil_image.fromarray(output)
output.save(sr_image_path)
```

图 3. 6 生成结果

3.3 测试结果

使用 PSNR 和 SSIM 对算法输出的超分辨率图与原图进行比较，由于模型只对图像的 y 通道做了计算处理，所以评估时只在 y 通道进行评估。测试结果如图 3.7 所示，效果不错。具体结果分析在第五节给出。

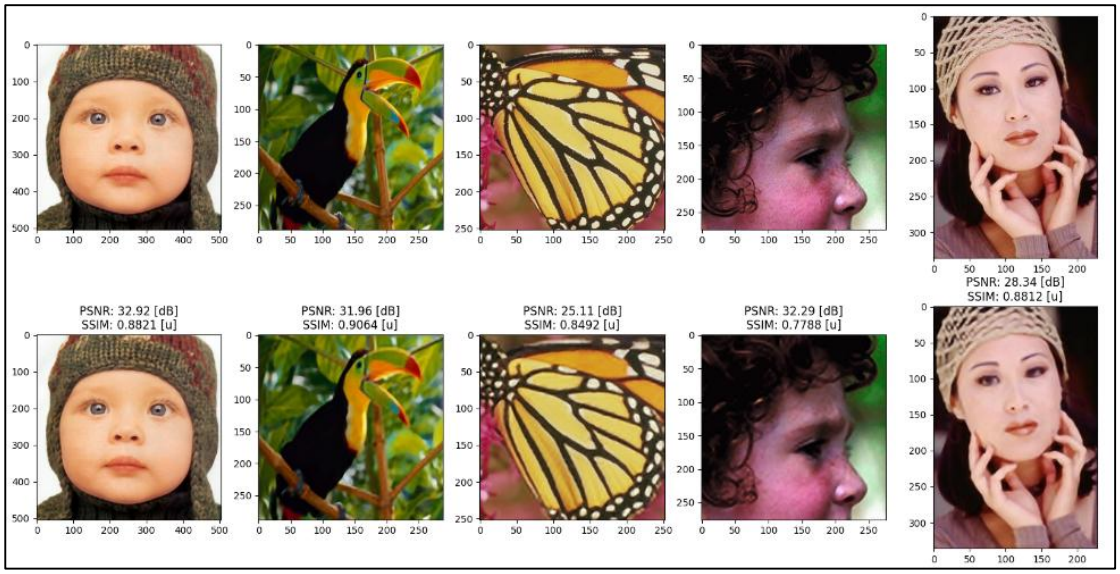

图 3. 7 SRCNN 测试结果

四．SRGAN

4.1 算法介绍

基于传统卷积神经网络的图像超分辨率算法一般使用插值或者 Padding 卷积的方式来实现。可以处理较小的放大倍数，但当放大倍数过大时则会导致生成的图像模糊缺乏细节。基于生成对抗网络的图像超分辨率算法(Super-Resolution Generative Adversarial Network，SRGAN)打破了传统模式，通过上采样和卷积实现图像超分辨率从而得到更加逼真的图像。其基本思想是将生成网络训练成包含权值和偏置参数的前向卷积神经网络，通过优化损失函数得到重构效果最好的生成网络。生成网络的训练过程是一个和判别网络博弈的过程，即利用训练好的判

别网络对生成网络生成的高分辨率图像和原图进行判别，如果判别网络分别不出原图和生成的高分辨率图像即认为生成网络已经训练到最佳状态，否则继续训练。采样这种方式，生成网络和判别网络在彼此的对抗过程中均可达到最佳的训练效果。最后使用生成网络计算出的超分辨率图像与原图相似。

SRGAN 网络由生成器(Generator)子网络和判别器(Discriminator)子网络组成。如图 4.1 所示，生成器子网络通过卷积层、激活函数层与数层残差网络层进行特征映射实现图像重建。判别器子网络网络结构如图 4.2 所示。通过判别器子网络来鉴别生成器子网络生成的图像是否与原图相似，当判别器子网络无法鉴别生成图像的真假时，整个网络即可适用于重建超分辨率图。生成器子网络负责实现超分辨率图像重建任务，判别器子网络负责将生成的超分辨率图像与原图像进行对比、识别与判断，并将判断结果返回给生成器子网络，生成器子网络根据返回值优化网络权重。

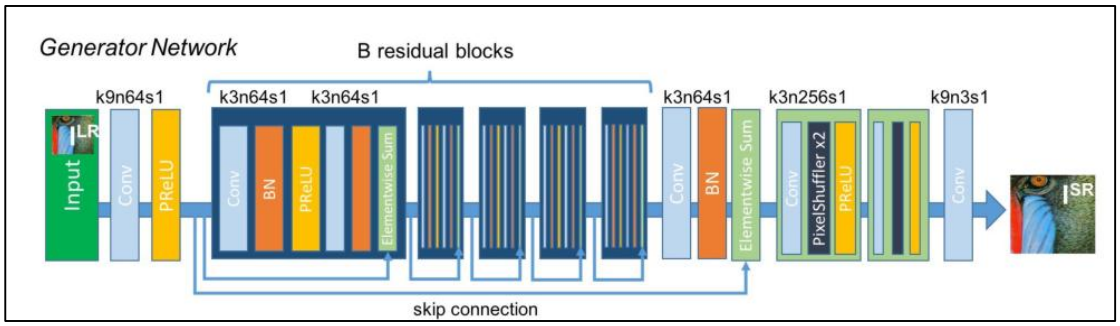

图 4. 1 生成器子网络结构图

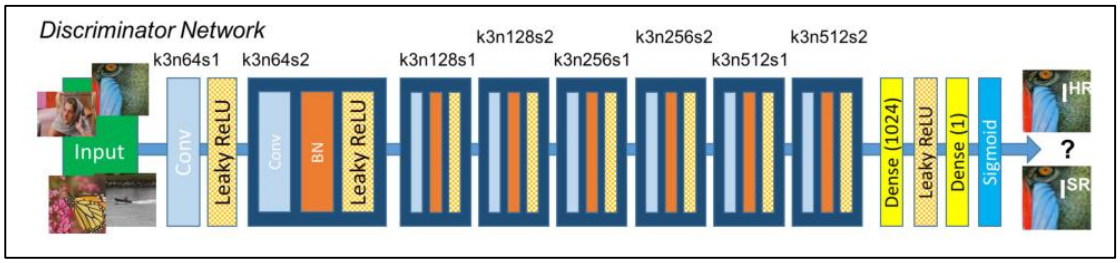

图 4. 2 判别器子网络结构图

不同于 SRCNN 使用均方差误差(MSE)作为损失函数训练网络，如式 4.1 所所示 SRGAN 的生成器子网络使用内容损失和对抗损失训练网络。MSE 虽然能够获得很高的峰值信噪比，但是恢复的图像会失去高频细节变得平滑，无法提供良好的视觉体验。所以 SRGAN 引入感知损失函数提高了超分辨率图像的质量，使其更加真实。

$$l^{SR} = \underbrace{\underbrace{l_X^{SR}}_{\text{内容损失}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{对抗损失}}}_{\text{感知损失}(VGG\ based)} \tag{4.1}$$

内容损失计算方式如式 4.2 所示，采用 VGG19 网络进行特征提取，在特征层面对生成图像和真实图像进行约束。

$$l_{VGG}^{SR} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}\left(\varphi_{i,j}(I^{HR})_{x,y} - \varphi_{i,j}\left(G_{\theta_G}(l^{LR})\right)_{x,y}\right)^2 \tag{4.2}$$

式中，$\varphi_{i,j}$表示第i个 max-pooling 层前的第j个卷积并经过激活层后的 feature map。$W_{i,j}H_{i,j}$表示描述了 VGG 网络内的各个特征图的尺寸。

对抗损失计算方式如式 4.3 所示，

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}\left(G_{\theta_D}(l^{LR})\right) \tag{4.3}$$

式中，$D_{\theta_D}\left(G_{\theta_D}(l^{LR})\right)$表示的是判别器认为生成图像为原图的概率。

4.2 关键实现代码

这一部分给出 SRGAN 在 Set5 数据集上测试的关键实现代码。其中包括模型的搭建与权重加载，数据预处理与测试结果生成。

- 模型搭建：将原图下采样得到的低分辨率图像作为输入，输入生成器子网络中便可得到生成的高分辨率图像，所以只对生成器模型进行代码实现。生成器是在 ResNet 的基础上实现，包含了多个残差块，如图 4.3 所示，每个残差块中包含两个 3×3 的卷积层，卷积层后接批规范化层(BN)并采用 ReLU 作为激活函数。

```
# 残差块 k3n64s1
class _ResidualConvBlock(nn.Module):
    def __init__(self, channels: int) -> None:
        super(_ResidualConvBlock, self).__init__()
        self.rcb = nn.Sequential(
            nn.Conv2d(channels, channels, (3, 3), (1, 1), (1, 1), bias=False),
            nn.BatchNorm2d(channels),
            nn.PReLU(),
            nn.Conv2d(channels, channels, (3, 3), (1, 1), (1, 1), bias=False),
            nn.BatchNorm2d(channels),
        )
    def forward(self, x: Tensor) -> Tensor:
        identity = x
        out = self.rcb(x)
        out = torch.add(out, identity)
        return out
```

图 4.3 残差块代码实现

上采样模块代码实现如图 4.4 所示，使用两个 2×亚像素卷积层(sub-pixel

convolution layers)来增大特征尺寸，使其输出图像尺寸与原图一致。

```
class _UpsampleBlock(nn.Module):
    def __init__(self, channels: int, upscale_factor: int) -> None:
        super(_UpsampleBlock, self).__init__()
        self.upsample_block = nn.Sequential(
            nn.Conv2d(channels, channels * upscale_factor * upscale_factor, (3, 3), (1, 1), (1, 1)),
            nn.PixelShuffle(2),
            nn.PReLU(),
        )

    def forward(self, x: Tensor) -> Tensor:
        out = self.upsample_block(x)
        return out
```

图 4.4 上采样模块

图 4.5 所示其他卷积层实现代码。如图 4.1 所示，包含三个卷积层，k9n64s1，k3n64s1 和 k9n3s1。

```
# Low frequency information extraction layer k9n64s1
self.conv1 = nn.Sequential(nn.Conv2d(in_channels, channels, (9, 9), (1, 1), (4, 4)),
    nn.PReLU(),)
# High-frequency information linear fusion layer k3n64s1
self.conv2 = nn.Sequential(
    nn.Conv2d(channels, channels, (3, 3), (1, 1), (1, 1), bias=False),
    nn.BatchNorm2d(channels),)
# reconstruction block k9n3s1
self.conv3 = nn.Conv2d(channels, out_channels, (9, 9), (1, 1), (4, 4))
```

图 4.5 卷积层代码实现

- 权重加载：与 SRCNN 一样，使用在 ImageNet 上预训练好的生成模型进行测试，如图 4.6 所示。同样使用 GPU 进行加速计算，将模型参数加载进模型中即可。

```
device = torch.device("cuda", 0)
g_model_weights_path = './checkpoints/SRGAN_x4-ImageNet-8c4a7569.pth.tar'

g_model = srresnet_x4(in_channels=3, out_channels=3,channels=64,num_rcb=16)
g_model = g_model.to(device)

# Load the super-resolution bsrgan_model weights
checkpoint = torch.load(g_model_weights_path, map_location=lambda storage, loc: storage)
g_model.load_state_dict(checkpoint["state_dict"])
g_model.eval()
```

图 4.6 模型权重加载

- 数据预处理与生成结果：与 SRCNN 不同，SRGAN 的生成模型是在低分辨率图像的 RGB 通道上进行计算处理，输入图像通道数为 3，最后输出生成的高分辨率 RGB 图像。在测试阶段，仅需读入图像转换成模型要求输入的 Tensor 格式，输入模型获得输出，最后将 Tensor 格式输出转换到 RGB 图像即可。这里不在给出代码实现。

4.3 测试结果

测试结果如图 4.7 所示。为了与 SRCNN 进行比较，所以同样只在颜色空间 YCbCr 的 y 通道上进行评估。具体结果分析在第五节给出。

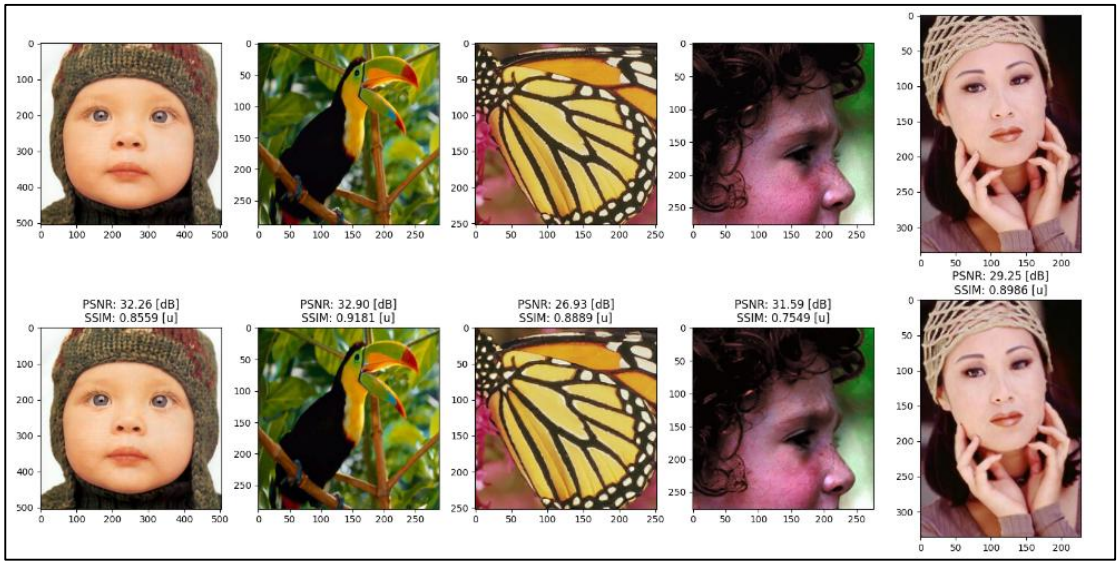

图 4. 7 SRGAN 测试结果

五．对比分析

5.1 训练过程

SRCNN 是以卷积神经网络为基础进行实现，采用端到端的监督学习方法。使用成对的低分辨率图像和高分辨率图像作为训练数据，以最小化生成的高分辨率图像与原始高分辨率图像之间的均方误差作为优化目标，基于逐像素损失进行训练。训练过程中，首先需使用传统的超分辨率算法(如插值法)将低分辨率图像初步转换成高分辨率图像，再通过多个卷积层进行特征提取和非线性变换，在损失函数的指导下，使得高分辨率图像在像素值上不断逼近原始高分辨率图像。

SRGAN 是基于生成对抗网络(GAN)的超分辨率重建方法。它由两个神经网络组成：一个生成器网络和一个判别器网络。生成器网络学习从低分辨率图像生成高分辨率图像，而判别器网络则尝试将生成的图像与真实高分辨率图像区分开来。在训练过程中，生成器网络和判别器网络交替进行训练，生成器网络尝试生成更逼真的高分辨率图像，而判别器网络则尝试准确地区分真实和生成的图像。采用这种方式，生成网络和判别网络在彼此的对抗过程中均可达到最佳的训练效果。生成器网络的训练过程是一个和判别器网络博弈的过程，即利用训练好的判别器网络对生成器网络生成的高分辨率图像和原图进行判别，如果判别器网络分

别不出原图和生成的高分辨率图像即认为生成器网络已经训练到最佳状态，否则继续训练。这样，生成器网络可以不断地学习如何生成更逼真的高分辨率图像，以欺骗判别器网络，最后使用生成网络计算出的超分辨率图像与原图相似。

在训练过程中，不同于 SRCNN 使用的 MSE 作为损失函数，SRGAN 的生成器子网络使用内容损失和对抗损失训练网络。使生成的高分辨率图像可以保留高频细节内容，提高了超分辨率图像的质量，使其更加真实。在图像预处理阶段，没有采用传统的超分辨率方法生成初步的高分辨率图像，而是在网络中使用 PixelShuffle 层进行图像上采样，使得生成的图像分辨率与原高分辨率图像一致。

5.2 结果对比

Bicubic、SRCNN 和 SRGAN 三种图像超分辨率算法在 Set5 数据集上的平均测试结果如表 5-1 所示。测试结果均采用 4 倍放大在颜色空间 YCbCr 的 y 通道上进行测试。可见平均测试结果 SRGAN 表现最好，但其网络复杂性和训练时间等开销也是最高。

表 5- 1 三种方法测试结果

	Bicubic	SRCNN	SRGAN
PSNR(dB) ↑	28.39	30.13	30.58
SSIM(u) ↑	0.8110	0.8595	0.8633

Bicubic、SRCNN 和 SRGAN 三种图像超分辨率算法在 Set5 数据集每张图像上的测试结果如图 5.1——图 5.5 所示。在每张比较图中，都可以观察到从左到右生成的高分辨率图像呈现从模糊到清晰的趋势。证明 SRGAN 可以获得最好的结果。图 5.1 和图 5.3 所示，SRCNN 得出的结果比 SRGAN 好，证明了 MSE 虽然能够获得很高的峰值信噪比，但是恢复的图像会失去高频细节变得平滑。而 SRGAN 在生成器网络的训练过程中，引入了内容损失，以 vgg 等网络计算出的图像高维特征作为指导，可以保留图像细节内容，提供良好的视觉体验。

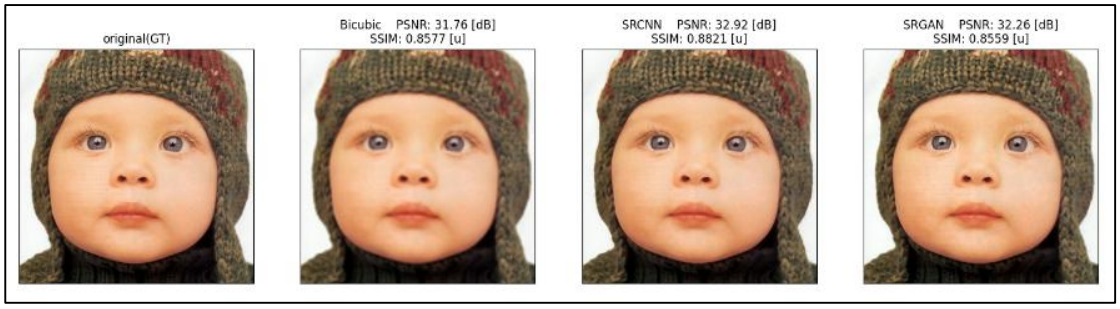

图 5. 1 baby

图 5. 2 bird

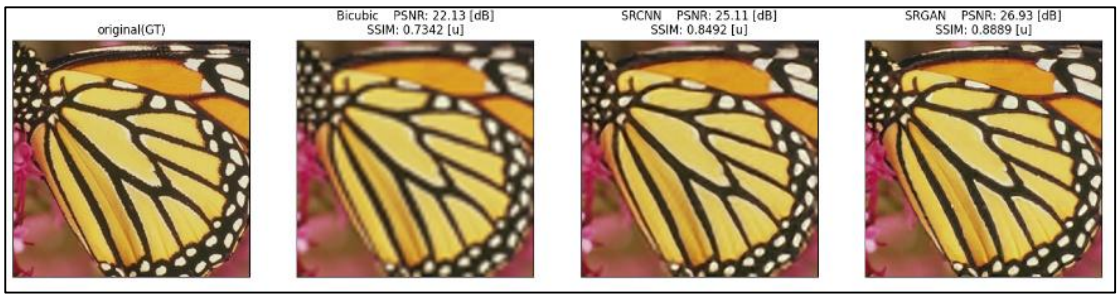

图 5. 3 butterfly

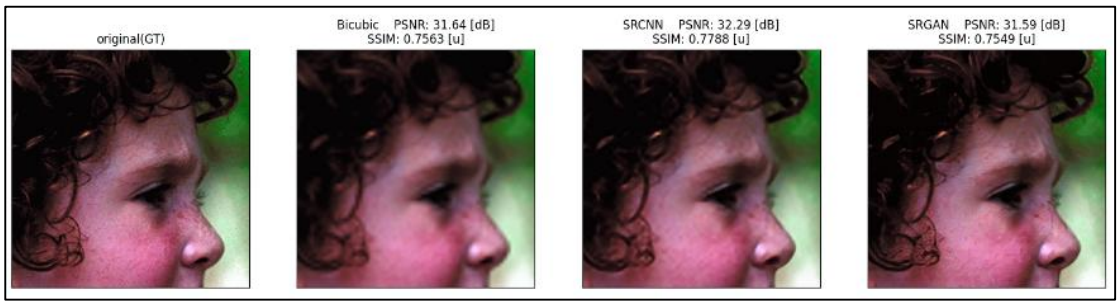

图 5. 4 head

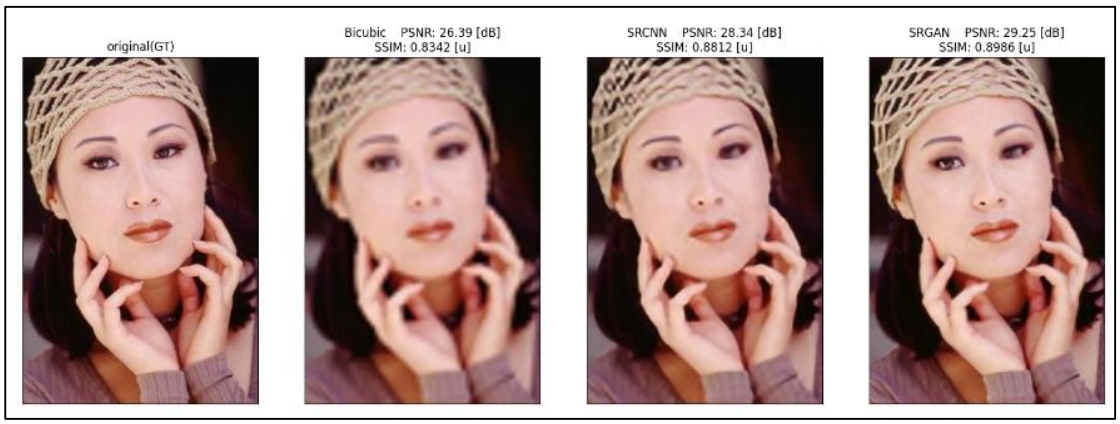

图 5. 5 woman

六．实验分析与总结

本次实验实现了基于逐像素损失的图像超分辨率算法 SRCNN 和基于 GAN 的图像超分辨率算法 SRGAN 在 Set5 数据集上的测试，并对比了两种类型的图

像超分辨率方法在训练过程和生成图像质量上的不同，完成了实验内容。

实验过程中，使用相同的下采样倍数得到 Set5 数据集的下采样数据。搭建了两个网络模型，分别为 SRCNN 和 SRGAN 的生成器子模型，使用预训练好的模型参数在下采样的 Set5 数据集上进行测试，生成高分辨率图像。

在测试阶段，使用 PSNR 和 SSIM 两种评价指标来衡量这两种方法与传统方法 bicubic 的生成质量。实验结果表明，SRGAN 在生成图像的视觉效果方面优于 SRCNN 和 bicubic。此外，还进行了图像可视化展示，并使用人眼观察评估了两种方法生成的图像质量。结果显示，SRGAN 生成的图像具有更好的细节保留能力和更高的视觉感知质量，更加清晰自然，具有更高的真实感，相比之下 SRCNN 生成的图像则显得较为平滑，缺乏真实感。

总的来说，SRCNN 和 SRGAN 是两种常用的图像超分辨率算法，分别采用不同的网络结构和训练方式，因此在训练过程和生成质量上存在一定的差异。实验结果表明，SRGAN 在生成高质量图像方面表现更好，但需要更长的训练时间和更复杂的网络结构，而 SRCNN 虽然训练速度快，但在保留图像细节方面表现相对较差。