

# 人工智能 Project2 图像修复

魏天昊 3150104461

2018 年 6 月 25 日

## 1 项目简介

### 1.1 开发环境

Python 版本为 3.5，深度学习框架使用的是 TensorFlow。

### 1.2 问题描述

目标是给定一张加入随机噪声的图片，恢复出原始图像。为了评估各种方法的优劣，我们需要自己构建数据和评测函数进行评估。首先生成随机噪声。

```
In [2]: generate_eval_data(0.4)
        generate_eval_data(0.6)
        generate_eval_data(0.8)
```

## 2 具体实现

在本项目中，我实现了两种方法，均取得了较好的效果。

### 2.1 KNN

首先实现的是 KNN 算法，作为 baseline。KNN 算法即选取缺失像素最近的几个点，根据距离进行加权求和作为该点的颜色。在图像修复问题中，KNN 是一个非常自然的想法。

KNN 的加权函数我使用了常用的距离倒数： $w = \frac{1}{\text{Euclidean Distance}}$ 。

为了选取最合适的邻近点数量，我评测了邻近点数量从 1 到 10 的表现。

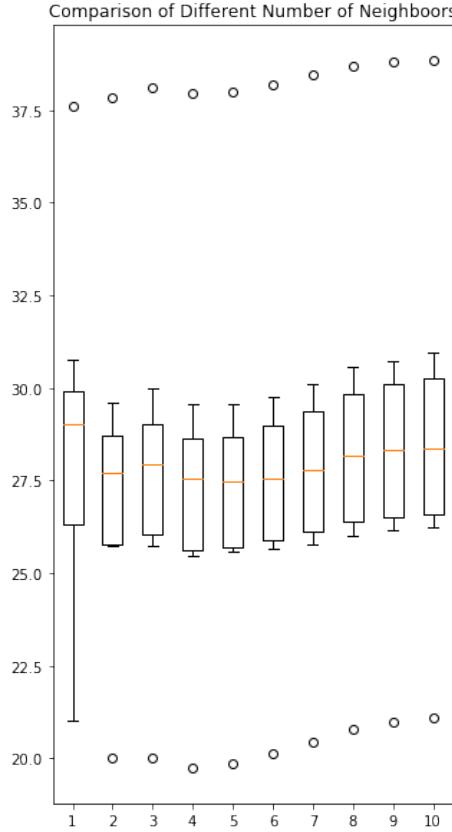
```
In [3]: plot_data = []
for choose in range(1,11):
    scores = evaluate(0.6, KNN, (choose,))
    plot_data.append(scores)

fig = plt.figure(1, figsize=(5, 10))
```

```

ax = fig.add_subplot(111)
bp = ax.boxplot(plot_data)
plt.title('Comparison of Different Number of Neighbors')
plt.savefig('knn_param')

```



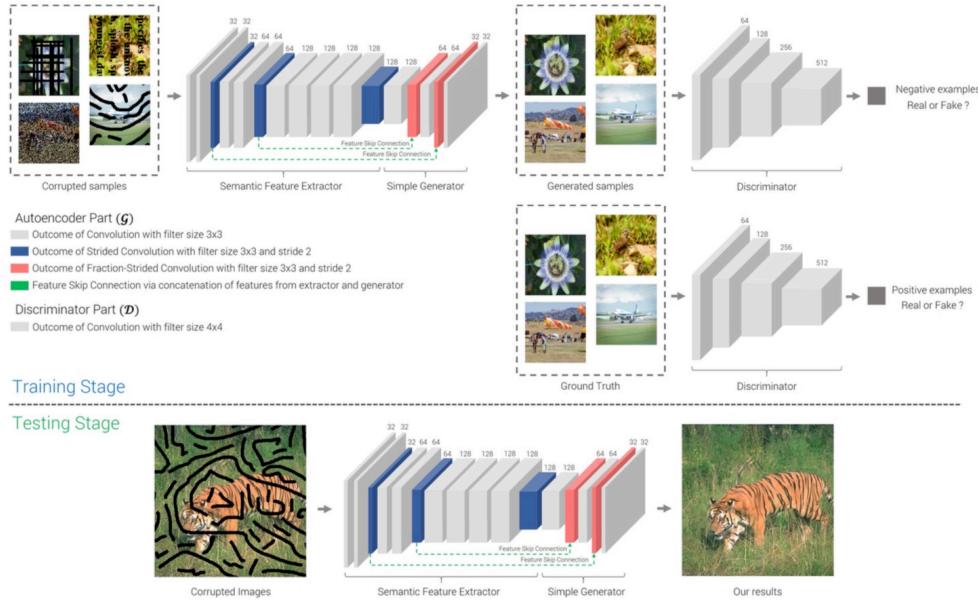
由上图可以看出，临近点为 4-5 时效果最好，因此之后均取值为 4。

KNN 算法虽然表现不错，但仍有一些缺点。1. 最近邻距离非常远时效果很差（常出现在噪音比例非常高时）。2. 只关注色彩信息，没有结构信息，边缘会出现模糊。3. 只关注局部信息，没有针对图片全局的优化。为了克服这些缺点，我使用了一种基于 GAN 的方法。

## 2.2 General Deep Image Completion

### 2.2.1 网络结构

该方法是基于 GAN 实现的图像补全算法。该网络包含生成器  $\mathcal{G}$  和判别器  $\mathcal{D}$  两个部分， $\mathcal{G}$  负责把缺失的图像补全为完整的图像， $\mathcal{D}$  负责判断生成的图像是否真实。网络结构如下：



$\mathcal{G}$  包含两个部分，语义提取及图像生成。可以注意到该网络的特点是非对称的 encoder-decoder 结构，在非对称设计中，decoder 更容易训练，能够更好的保留图像语义信息。同时 skip layer 也提高了图像细节的保留度。

## 2.2.2 损失函数

**Reconstruction Loss:** 针对  $\mathcal{G}$  的生成效果，用于衡量与 ground truth 之间的差别。使用了  $L1$  loss， $L1$  loss 相比于  $L2$  loss 能够产生更加真实的效果，因为  $L2$  loss 对于误判较大的像素惩罚力度更大，因此会更倾向于产生更模糊的图片。而在本次项目中，我们最终使用的是  $L2$  loss 来衡量图片生成结果。这导致了虽然该方法修复的图像十分真实，在视觉效果上与 KNN 相比具有显著的优势，但 loss 相比于 KNN 却没有明显的降低。

$$\mathcal{L}_{rec} = \frac{1}{H \times W} \sum_x \sum_y ||I_g t(x, y) - \mathcal{G}(I)(x, y)||_1$$

**Adversarial Discriminator Loss:** 该方法也采用了对抗训练，该项 loss 用于使判别器能够更好地分辨出真实图像和构造图像。从而为  $\mathcal{G}$  提供能够欺骗  $\mathcal{D}$  的梯度。

$$\mathcal{L}_{adv_D} = \frac{1}{2} \times ||\mathcal{G}(I_g t) - 1||_2 + \frac{1}{2} \times ||\mathcal{D}(\mathcal{G}(I)) - 0||_2$$

**Adversarial Generator Loss:** 该项 Loss 用于使生成器能够生成更加接近 ground truth 的图像。

$$\mathcal{L}_{adv_G} = \frac{1}{2} \times ||\mathcal{D}(\mathcal{G}(I)) - 1||_2$$

综上所述，最终的优化目标为：

$$\mathcal{G}^* = \arg \min_{\mathcal{G}, \mathcal{D}} \lambda_{\mathcal{G}} \mathcal{L}_{rec} + \lambda_{\mathcal{D}} \mathcal{L}_{adv_G} + \lambda_{\mathcal{D}} \mathcal{L}_{adv_D}$$

### 3 实验结果

#### 3.1 KNN 效果可视化

以噪音比例为 0.6 时为例。

```
In [4]: visual_sample(0.6, KNN, (4,));
```



#### 3.2 Deep Image Completion 效果可视化

以噪音比例为 0.6 时为例。

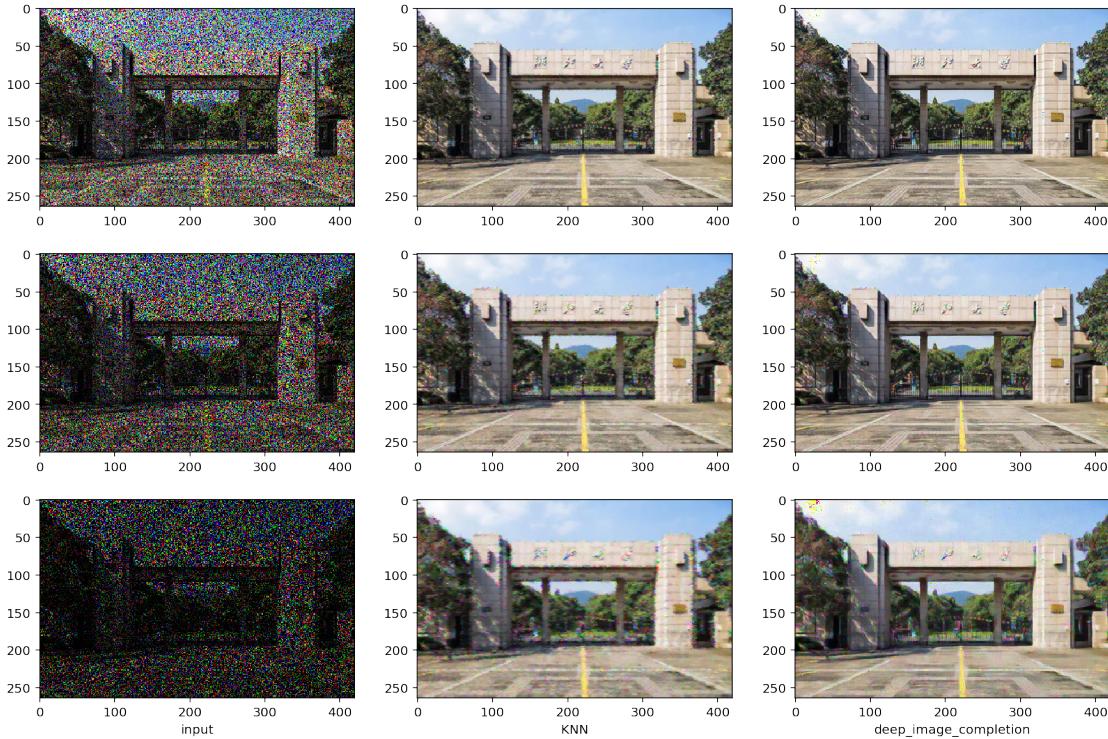
```
In [5]: visual_sample(0.6, deep_image_completion,());
```



### 3.3 KNN 与 Deep Image Completion 效果对比

以下三幅图的噪音比例分别为 0.4、0.6、0.8。

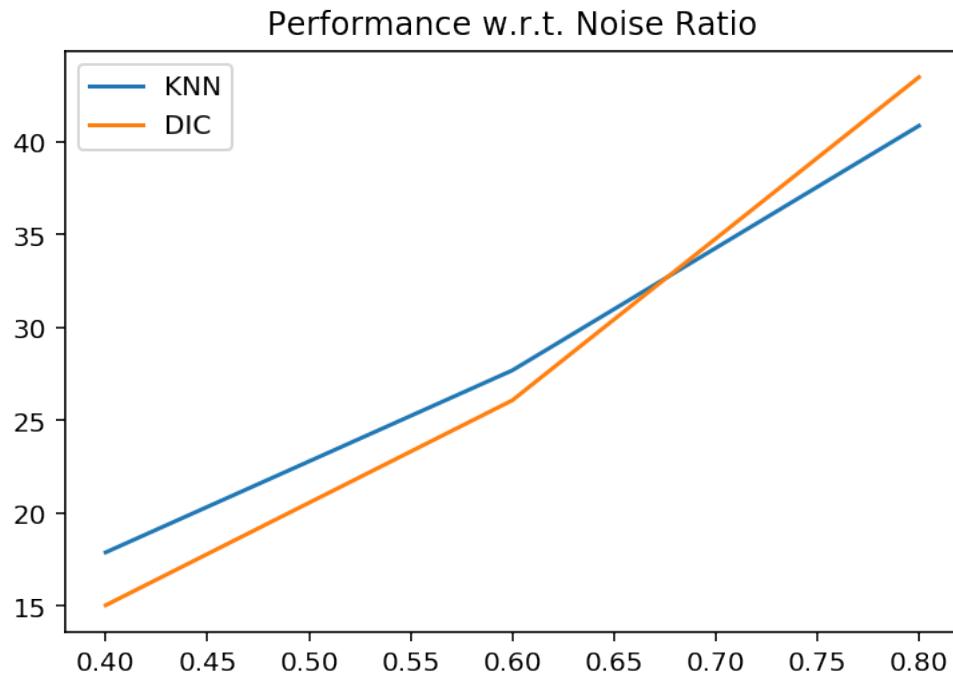
In [6]: `compare(KNN,(4,),deep_image_completion,());`



### 3.4 KNN 与 Deep Image Completion 数值结果比较

```
In [7]: x = [0.4, 0.6, 0.8]
knn_data = []
gan_data = []
for ratio in x:
    knn_data.append(np.mean(evaluate(ratio, KNN, (4,))));
    gan_data.append(np.mean(evaluate(ratio, deep_image_completion, ())));
plt.plot(x, knn_data, label='KNN')
plt.plot(x, gan_data, label='DIC')
plt.legend()
plt.title('Performance w.r.t. Noise Ratio')
```

Out [5]: <matplotlib.text.Text at 0x11d51b6d0>

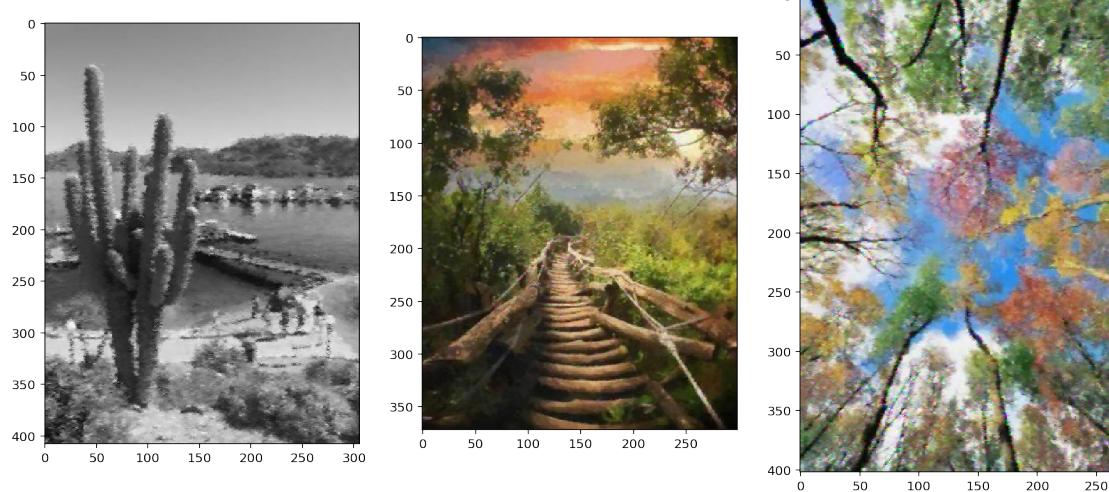


可以看出虽然 Deep Image Completion 视觉效果显著高于 KNN，但数值结果并没有显著优势，甚至还在某些情况下略低。原因应该如前文所述，是评测用 Loss 与训练用 Loss 不同所致。训练所使用的 Loss 可以让图片看起来更真实。

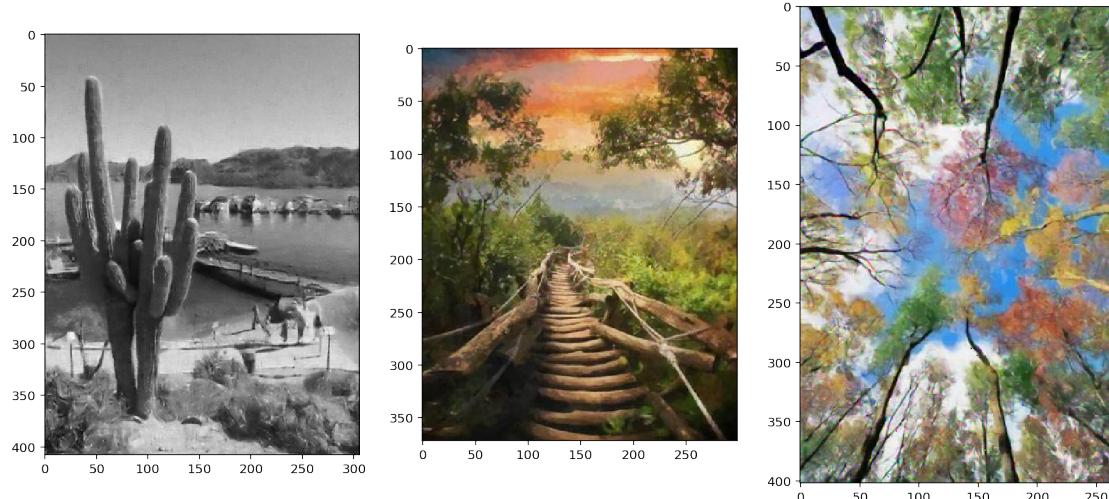
## 4 生成 Test 图像

最后我们生成两种方法对应的测试图像，并选取 Deep Image Completion 的结果作为最终结果提交。

In [8]: `test(KNN,(4,))`



```
In [9]: test(deep_image_completion,())
```



## 5 参考资料

Ching-Wei Tseng, Hung-Jin Lin, Shang-Hong Lai, General Deep Image Completion with Lightweight Conditional Generative Adversarial Networks, Proceedings of the British Machine Vision Conference