

# 浙江大学实验报告

**ImgRecovery**

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **专业班级** | **姓名** | **性别** |
| **3150104669** | **混合1505** | **张子健** | **男** |

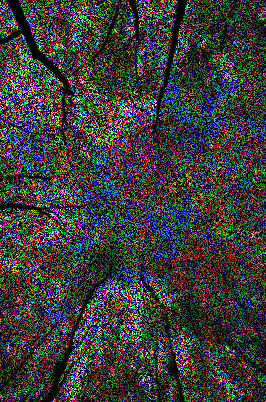
指导教师： 吴飞

电子邮件地址： 3150104669@zju.edu.cn

实验日期： 2018 年 5 月 12 日

## 实验目的与要求

1. 给定3张受损图像，尝试恢复他们的原始图像。

1. 原始图像包含1张黑白图像（A.png）和2张彩色图像（B.png, C.png）。
2. 受损图像（）是由原始图像（）添加了不同噪声遮罩（noise masks）（）得到的（），其中是逐元素相乘。
3. 噪声遮罩仅包含{0,1}值。对应原图（A/B/C）的噪声遮罩的每行分别用0.8/0.4/0.6的噪声比率产生的，即噪声遮罩每个通道每行80%/40%/60%的像素值为0，其他为1。
4. Code：目前提供的实现代码，利用逐行回归实现图像恢复，效果一般。鼓励大家开发更有效的恢复方法。
5. Data：提供的三幅待恢复图像，提供给大家。要求设计恢复算法，得到恢复图像，和实验报告一起提交。
6. 程序语言必须使用Python。

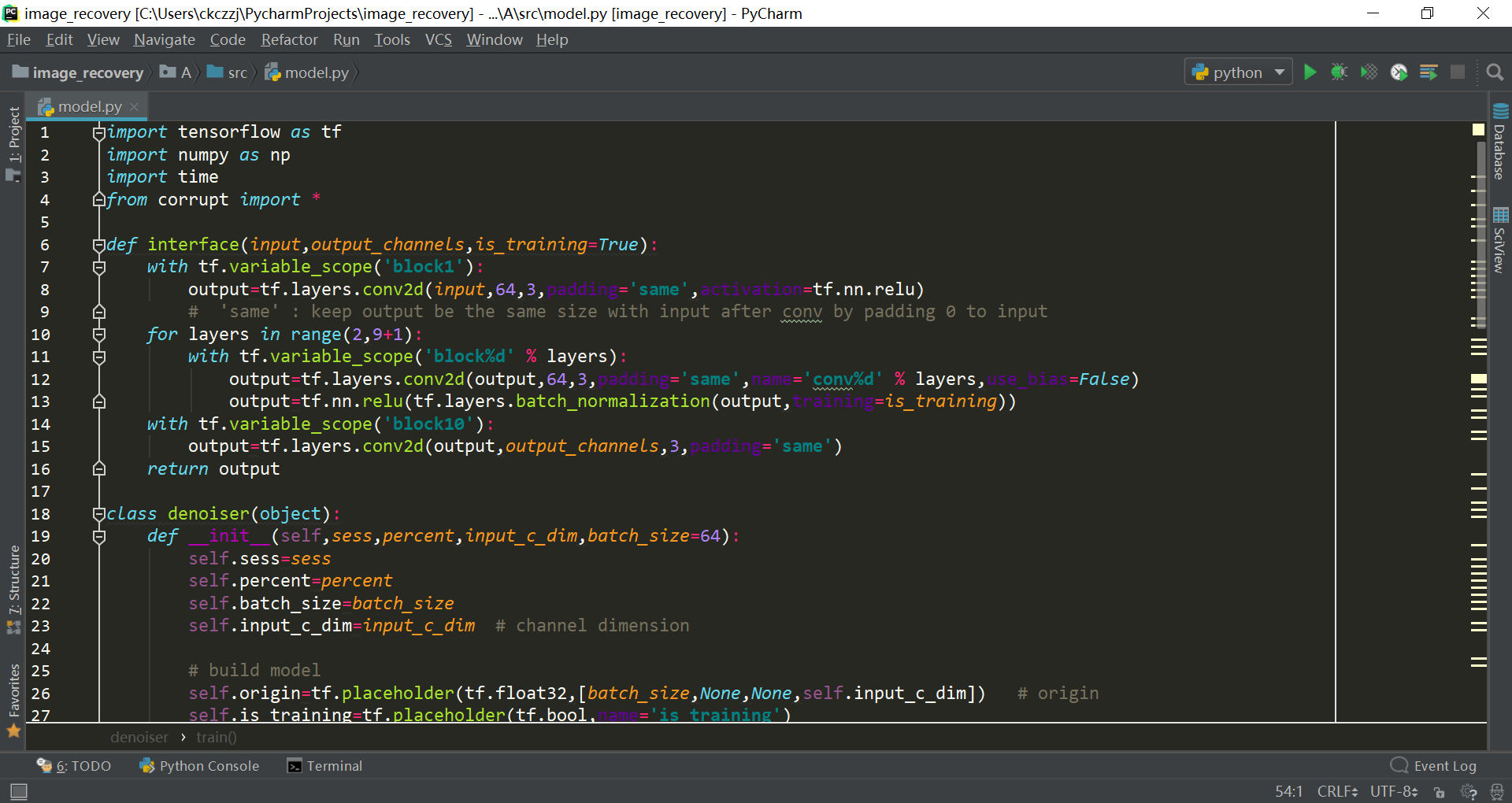
## 实验环境与分工

1. 开发环境

PC Windows 10 Jet Brains PyCharm免费教育版

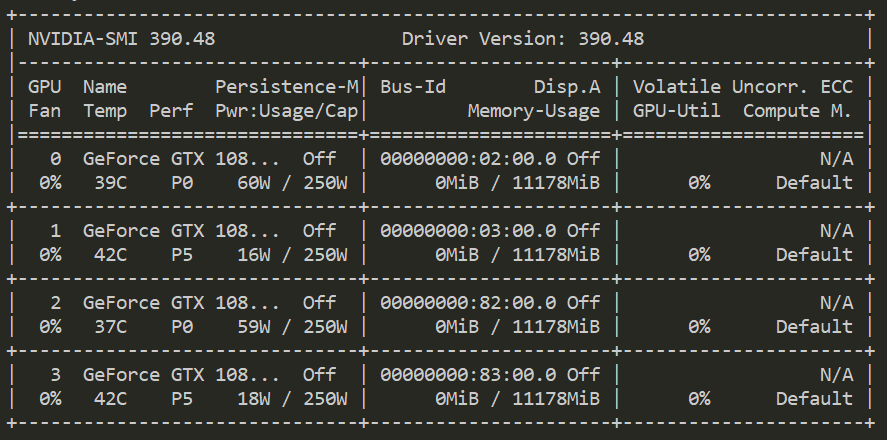
PYTHON\_VERSION="2.7"

主要使用的库有numpy和tensorflow，以及图片的读写库scipy.misc



1. 运行环境

GPU服务器，由4块GTX 1080 GPU进行并行计算。



本次实验所有工作均由我本人独立完成。

## 实验内容

1. **深度学习**

计算机视觉和自然语言处理是当下深度学习的两个主要研究和应用领域。

Alex在2012年提出的AlexNet网络结构模型引爆了神经网络的应用热潮，并赢得了2012届ImageNet(图像识别大赛)的冠军，使得CNN成为在图像分类上的核心算法模型。

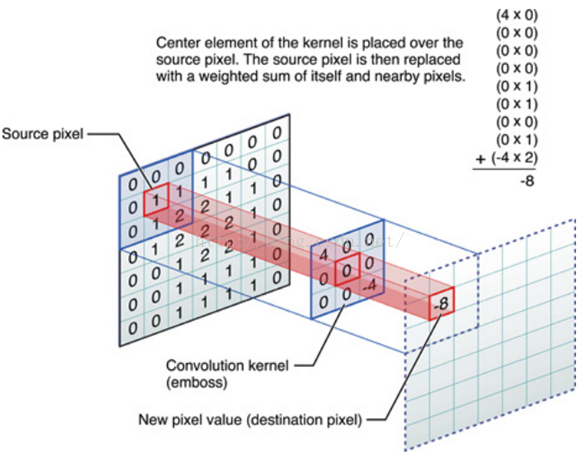
在图像处理方面，CNN对于传统机器学习的优势在于，它能自动提取图像的特征并进行学习，而不是局限于人为的特征输入。

所以本次实验我选择使用CNN完成。

大多数的深度学习任务要考虑的问题有网络架构、目标函数、训练技巧以及训练集等，所以我将从这几个方面介绍我的实验方法。

1. **网络架构**

我的网络架构以CNN为基础，CNN的示意图如下：



我的网络架构是一个n层(层数可变，n作为一个超参数)CNN网络，隐藏层都使用64个filter，卷积核大小都为3\*3，激活函数都为relu，为了保持图像大小不变，在每层CNN运算前，都会对前一层的输出做padding 0的操作。输出层使用input\_channel(输入图像的通道个数)个filter，卷积核大小仍为3\*3，输出恢复图像。

上述的CNN层结构被tensorflow封装为函数，可以直接调用。

1. **Loss Function**

由于实验中给出了测评结果的函数，所以我直接使用各通道所有像素差值的二范数的和作为损失函数进行训练。

1. **训练数据集**

我的训练方法也很简单，将训练集中的无噪声的图像输入网络，网络自动按照本实验的加噪方法对图像进行加噪，然后将加噪图像输入网络，得到输出后与原来的无噪声的图像计算损失，由tensorflow自动优化网络参数，最小化损失函数。

显然，由于每次网络会自动对输入图像进行加噪(即按比例随机置0)，所以训练数据几乎是无穷的且不同的。

我选取了一个黑白图像数据集(400张180\*180\*1的黑白图片)训练恢复黑白噪声图像，该数据集多用于图像处理领域。对于RGB图像，由于找不到合适的风景图数据集，所以我使用了Cifar10数据集，从中选取了3000张32\*32\*3的图像训练恢复RGB噪声图像。

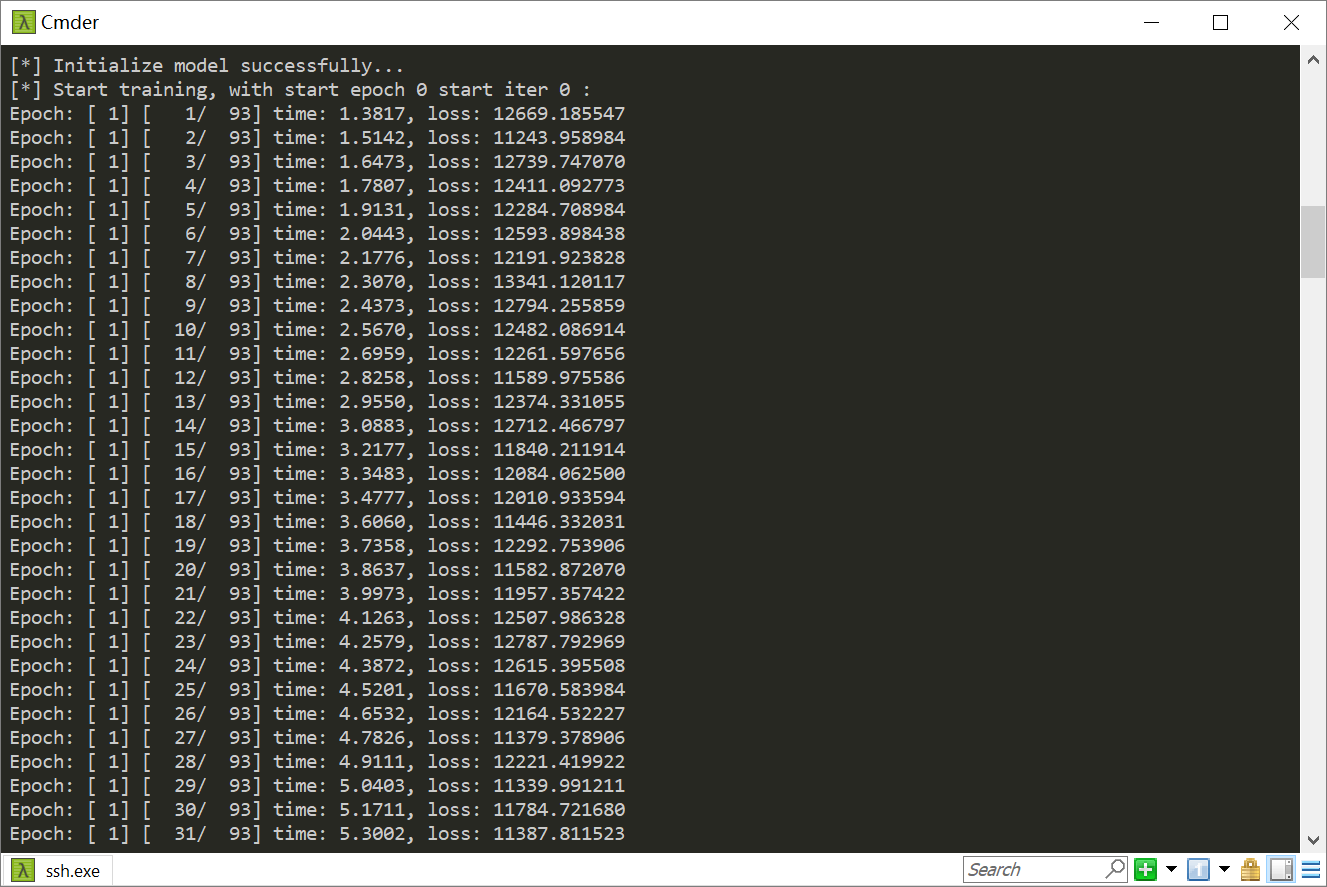
1. **训练技巧**

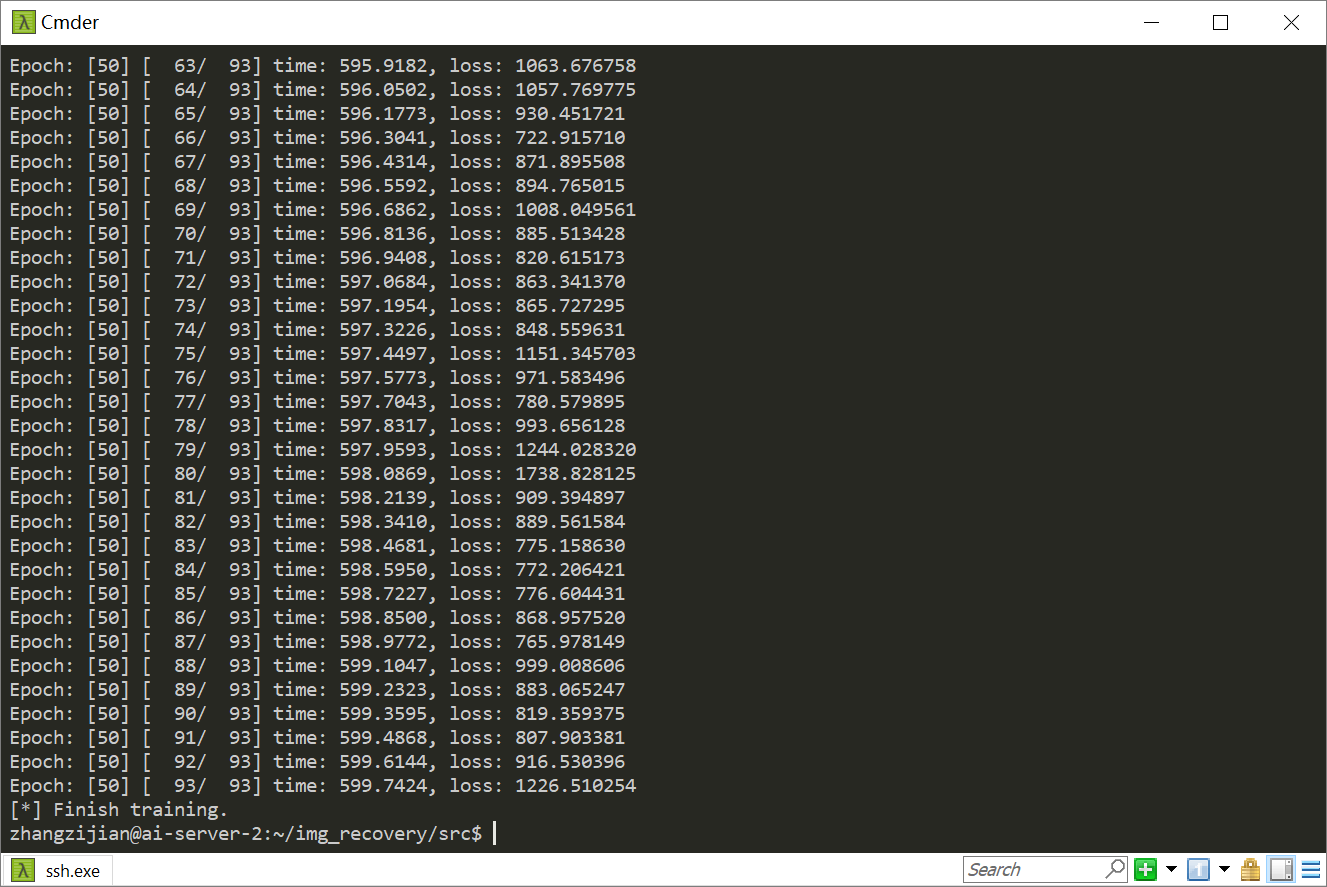
训练中我使用了batch normalization的技巧，这是2015年Google发表的论文《Batch Normalization: Accelerating Deep Network Training by  Reducing Internal Covariate Shift》中提到的算法，它一定程度上能解决梯度弥散、梯度爆炸、过拟合等问题，还能加速神经网络的训练。

在深度网络的训练中，每一层网络的输入都会因为前一层网络参数的变化导致其分布发生改变，这就要求我们必须使用一个很小的学习率和对参数很好的初始化，但是这么做会让训练过程变得慢而且复杂。BN在每一层的激活函数前对数据进行归一化来解决这个问题。

## 实验结果

1. **训练过程**





1. **训练结果**

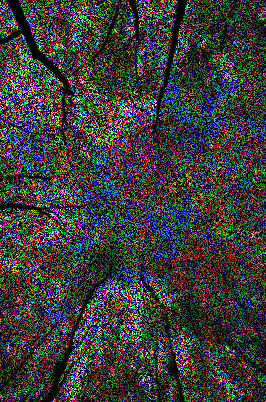
对于A.png，使用32的batch\_size，40的epoch，8层隐藏层CNN，80%的加噪率，结果如下：

对于B.png，使用32的batch\_size，50的epoch，10层隐藏层CNN，40%的加噪率，结果如下：

 ****

对于C.png，使用32的batch\_size，50的epoch，10层隐藏层CNN，60%的加噪率，结果如下：

## 实验不足与改进

代码中未实现Saver的checkpoint，导致模型无法保存，每次需要重新训练。

由于时间有限，训练集选的不是很好，如果能选点风景图作为训练集，效果可能会更好。

没有实现验证集和测试集，因为图片大小不一，所以测试loss也没什么意义。

未对超参数做一些其它的尝试，对比不同超参数的效果。

网络结构过于简单，只有简单的几层CNN，未来可以考虑将mask作为一个输入进行训练。

## 参考文献

算法参考：

Alex et al.《ImageNet Classification with Deep Convolutional Neural Networks》

Google《Batch Normalization: Accelerating Deep Network Training by  Reducing Internal Covariate Shift》

编码参考：

Tensorflow官方文档： https://www.tensorflow.org

## 声明

本人张子健（3150104669）声明：本次实验从构思、设计到编码实现均由我本人独立自主完成，未参考其它任何去噪算法，未与其他任何人交流。所使用的IDE及编码环境均为免费开源的。