基于深度卷积网络的
图像去噪研究

Research on Picture Noise Reduction Based Deep Convolution Network

计算机科学与技术系

李京　　PB13209115

张信明　　教授

2017 年 5 月 19 日

中国科学技术大学

本科毕业论文

题 目 基于深度卷积网络的图像去噪研究

英 文 Research on Picture Noise Reduction Based Deep Convolution Network

院 系 计算机科学与技术学院

姓 名 李京

导 师 张信明

日 期 二〇一七年五月

致谢

首先要感谢我的指导老师张信明老师，在张老师的指导下，我完成了毕业论文的选题、修改，也非常感谢张老师能够支持我做自己非常感兴趣的课题。在此期间，他时常关注我们的进展，并给出指导意见。

在此，我还要感谢好朋友王悦同学，首先是他邀请我一起做深度学习方面的课题，而且在课题研究过程中遇到问题的情况下，他给了我很大的帮助和建议。

感谢大学四年所有的老师。在这里老师们不仅传授给了我们知识、开阔了我们的视野，更多的是让我们拥有了更强的自学能力、更多的挑战自我的自信、以及坚持下去的毅力。感谢大学期间陪伴过我支持过我帮助过我的每一个同学，正因为你们，大学生活才多姿多彩、欢声笑语。最后还要感谢生我养我支持我鼓励我的爸妈！

感谢所有曾经支持过我、陪伴过我的每一个人！

目录

中文内容摘要.....2
Abstract.....3

第一章 绪论及背景.....4
1.1 背景.....4
1.2 图像去噪的发展状况.....4
1.3 深度卷积神经网络.....5
1.4 本文章节安排.....6

第二章 相关工作.....7
2.1 图像的噪声模型.....7
2.2 深度神经网络.....7
2.3 卷积神经网络.....12
2.4 卷积神经网络的训练.....13
2.5 卷积神经网络应对图像处理.....14

第三章 深度卷积网络模型.....16
3.1 卷积神经网络基础.....16
3.2 深度卷积网络应用于图像降噪.....17
3.3 功能性拓展.....21

第四章 实验结果及分析.....24
4.1 训练数据.....24
4.2 测试数据及对比.....24
4.3 测试结果及分析.....25

第五章 全文总结及展望.....32

参考文献.....33
附录.....34

中文内容摘要

在这个信息时代,随着互联网的发展以及人工智能的发展,每个人都在习惯了利用自己的移动终端(手机、平板等)分享自己的生活,分享数字图像是其中一项很重要的内容。然而受到各种不可避免的外界影响，使得图像受到了噪声的干扰。图像的噪声不仅在一定程度上破坏了视觉感受体验，同时也对计算机对图像做进一步处理造成很大干扰。对于目前已经存在的一些去噪效果较好的算法，比如BM3D、LSSC，则又会有运算量较大的的问题。本文基于深度卷积网络，对图像进行去噪处理，并加以批量归一化等方法，以加速网络的训练过程。以高斯噪声为例，该网络对固定噪声水平的噪声以及不固定噪声水平的噪声，均有较好的去噪效果，同时拥有几十倍于 BM3D 的性能。同时，使用 GPU 加速将获得更高的性能。非常适合运算性能有限的移动端设备以及运算任务较多的服务器。

关键词：深度卷积神经网络（CNN），高斯噪声，图像去噪，批量归一化。

Abstract

In this information age, with the development of the Internet and the development of artificial intelligence, everyone is using their own mobile terminal to share their own lives, sharing digital images is one of the very important content. At the same time, the image processing is also a hot topic in artificial intelligence research. But by various unavoidable external influences, making the image subject to noise interference. Image noise not only affects the visual perception, but also on the computer to do further processing of the image caused great interference. There are some algorithms that have a good denoising effect, such as BM3D, LSSC, but they need a long time to run. In this paper, the image is denoised based on the deep convolution network, and the method of batch normalization is adopted to accelerate the training process of the network. Taking Gaussian noise as an example, the network has good denoising effect for noise at fixed noise level and noise with no fixed noise level, and it is several times faster than BM3D. Performance gains will be greater when using GPU acceleration. Very suitable for computing performance limited mobile terminal equipment and busy servers.

Keyword: Deep convolution network, Gaussian noise, Image denoising, Batch normalization。

第一章 绪论及背景

1.1 背景

随着互联网的发展以及各种便携式终端的普及，人们越来越多的在互联网上分享自己的生活，图像信息包含了较多的信息，成为生活中不可缺少的一部分。人类所获取外界信息中视觉信息占据了 80%，触觉、嗅觉等等占据了剩余的 20%，所以图像相对于文字更能够表达当下心情以及感受。

如今，数字图像已经在各个方面广泛应用在人们身边，几乎每一个人均有可以采集数字图像的终端设备，手机、相机等等各种产品的普及已经极大的改变了我们的生活。人们希望能够从图像中获得更多信息以及更好的视觉体验，然而由于某些不可避免的外界干扰（自然硬件、软件噪声以及非自身噪声影响），图像在采集、传输等过程中原始信息受到了破坏，出现了噪声。

噪声在一定程度行破坏了图像的视觉感受体验,同时,也会在一定程度上损坏图像所携带原始信息。在噪声严重情况下，可能导致图像不可辨认。而在自动识别、遥感、测绘等方面，噪声所造成的影响更大。所以当前图像处理的一个很重要的操作，即为图像去噪。我们使用图像去噪，从一个包含噪声的图像甚至原始信息已经有所丢失的情况下，尽可能还原其原始信息，这对于图像处理以及其他应用，图像的去噪较为重要。

当前去噪方法效果较为优秀的有 BM3D 等算法，算法最终效果较为优秀，对于当前更为流行的移动终端，会带来较大的运算压力。于是寻找一个去噪效果优秀且具有较高性能的去噪方法颇有意义。

1.2 图像去噪的发展状况

图像的去噪：从一个包含噪声、原始信息遭到部分破坏的图像中，尽可能地去除噪声，还原原始信息。这就要求，图像的去噪还隐含着额外的目标：图像中原有的纹理、边缘等等细节信息需要尽可能的保留，只有如此才能真正发挥图像去噪的作用。然而图像中部分细节信息和噪声信息是极为相似的，噪声中包含较多高频信息，同时，细节信息均属于高频信息，将其完全分离是很难实现的，故图像去噪的一大难点便是如何更好的保留细节信息。因此去噪的同时追求细节的保留是当前图像去噪算法的目标。

对于不同特点的噪声，现已出现很多图像去噪方法。对于图像去噪方法我们可以将其分为两类：空间域去噪、频域去噪。其中，空间域去噪去噪的方式为：直接修改图像的像素值。常见的空间域去噪有均值滤波降噪、中值滤波降噪、纳维滤波降噪等等；频域去噪亦称变换域去噪。频域去噪的方式为：从原始空间到频域，对图像进行变换处理，使图像呈现在频域，再对其进行处理，最后进行逆变换操作，至此完成降噪过程。常见的域变换方法有傅立叶变换、拉普拉斯变换等[7]。

近年来，一些新颖的去噪模型被提出，包括非局部自相似性模型(nonlocal self-similarity models，NSS)、梯度模型(gradient models)、马尔科夫随机场模型(Markov random field models, MRF)等。特别是 NSS 模型如 BM3D、LSSC 等方法目前很受欢迎。

尽管以上模型具有很高的去噪质量，然而他们尚存一些缺陷。这些方法在测试阶段通常面临着复杂的优化问题，绝大多数不得不牺牲计算效率以达到预想的效果而难以实现高性能。

1.3 深度卷积神经网络

深度学习(deep learning)是机器学习的分支，是机器学习中一种基于对数据进行表征学习的方法。人工神经网络的历史已经较为久远。

机器学习从最早的诞生到如今，我们可以认为其经历过两个发展阶段：浅层学习（shallow learning）和深度学习（deep learning）。上世纪 80 年代后期，反向传播算法（Back Propagation，BP）的出现以及应用极大的推进了机器学习的发展，基于统计的机器学习模型引发了新的浪潮。反向传播算法在神经网络算法中可以自动修正网络中的参数，使得网络的输出更大程度符合训练数据[15]。到如今，反向传播算法的应仍然十分广泛。90 年代以后，SVM（支持向量机，Support Vector Machine）最大熵法等更多的浅层机器学习模型出现在人们的视线中。这些模型框架大多包含一层隐藏结点或者不包含隐藏节点。1989 年，扬·勒丘恩（Yann LeCun）等人开始将 1974 年提出的标准反向传播算法应用于深度神经网络，这一网络被用于手写邮政编码识别。尽管算法可以成功执行，但计算代价非常巨大，神经网路的训练时间达到了 3 天，因而无法投入实际使用。而今，随着科技的进步以及计算性能的提升，神经网络以及深度学习再次引发研究的热潮。

深度卷积神经网络的起源来源于深度神经网络，相对于深度神经网络，深度卷积网络拥有局部连接、权值共享以及下连接的特征，在很多情况下更容易进行对特征信息的提取。在很多方面，深度卷积神经网络已经做出了自己的极大的贡献。

1.4 本文章节安排

本文第一章为背景绪论部分，第二章将讲述相关基础工作，第三章提出深度卷积网络的结构以及对其进行拓展的方式，第四章节将讲述对网络的训练以及对训练结果的展示，第五章为对全文的总结以及对未来的瞻望。

第二章 相关工作

2.1 图像的噪声模型

信号获取以及传输过程中，会受到外在能量所产生信号的干扰，频率、强弱变化无规律，杂乱无章。我们可以认为一个受噪声干扰的图像为原始信息与噪声信息的叠加。图像去噪的任务就是区分图像中的干扰信息，并将其去除。由于干扰源众多且不固定，噪声特性复杂且具有非常强的随机性。图像的噪声信息我们可以将其认为一个随机过程，利用概率统计的方法来分析噪声。那么绝大多数图像噪声服从绝对值为 0 的高斯分布：

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}} \tag{2-1}$$

其中式中x表示噪声中的灰度值，符合高斯分布，σ为噪声的标准差。那么一个受到噪声干扰的图像可以表示为：

$$Y = R + N \tag{2-2}$$

其中 Y 为观察到的包含噪声的图像，R 为不包含噪声的图像，N 为噪声。从该模型上看，该加性噪声仅与外界因素有关，与原始图像无关。如图 2.1 所示

图 2.1 图像的噪声模型

因此，在此模型下，我们可以描述图像的去噪算法为：输入信息为包含噪声的图像 Y，那么该算法的输出则为图像的原始信息 X。

2.2 深度神经网络

深度神经网络由多个网络层堆叠而成，每一层包含多个节点。神经网络的运算在节点中完成。每个节点对于输入的每个数据，使用一组系数（亦称权重）与之结合，作为该节点的数据或者经过激活函数后作为该点的数据。如下图所示

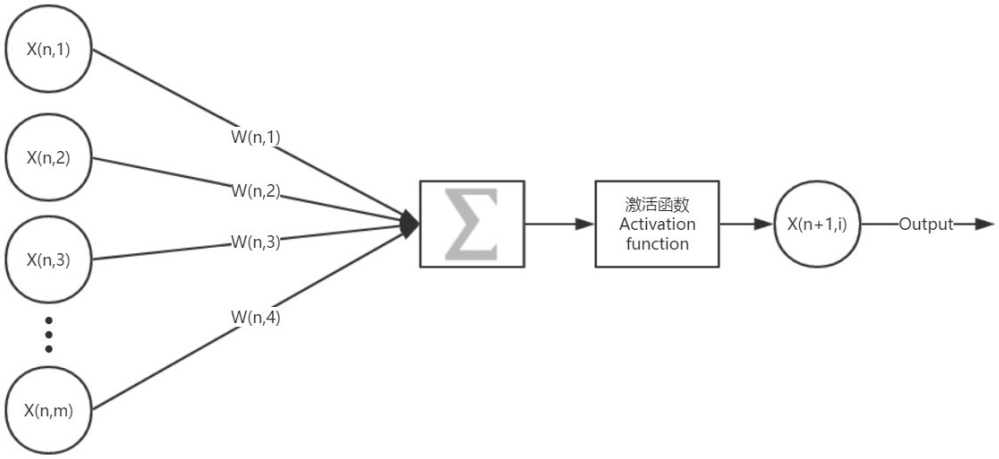


图 2.2 深度神经网络节点

$$Output = fun(\sum_{i=1}^{m} x_{n,i} \bullet w_{n,i}) \tag{2-3}$$

其中 $x_{n,i}$ 为第 n 层网络第 i 个节点的输出值，m 为第 n 层网络的节点的数量，fun 为激活函数。激活函数通常具有的特点有：非线性、可微性、单调性等，我们再网络中加入激活函数处理，使得网络中出现更多的非线性，继而神经网络能够获得更优的最终效果。

常用的激活函数有 ReLU、Sigmoid、tanh 等。

（1）ReLU：ReLU 近期较为常用，它的计算方式如下：

$$f(x) = \max(0, x) \tag{2-4}$$

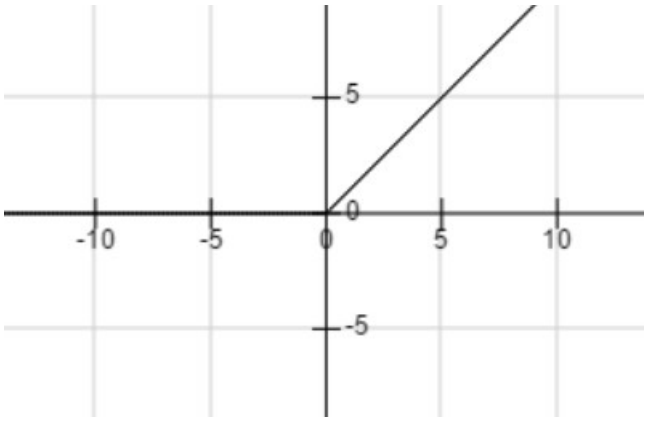


图 2.3 ReLU 激活函数的函数曲线

ReLU 激活函数的优点较为明显，它拥有较快的处理速度，而且在 x>0 情况下呈

线性关系，故在一定情况下拥有更快运算速度以及更快的收敛速度。缺点在于：当传入一个极大的梯度，那么当更新参数以后，该节点对其他数据将失去作用。

（2）Sigmoid：Sigmoid 激活函数也是一个非常常用的激活函数。它的数学表达式为：

$$f(x)=\frac{1}{1+e^{-x}} \tag{2-5}$$

Sigmoid 能够将实数集映射到 0 到 1 之间，函数图像如图 2.4 所示

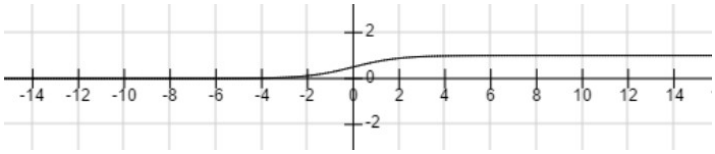


图 2.4 Sigmoid 激活函数的函数曲线

Sigmoid 拥有两个明显的优点：Sigmoid 求导较为容易，更加方便运算。其次，Sigmoid 的值域为(0,1)，较为稳定，十分适合用作输出层。不过遗憾的是 Sigmoid 容易出现梯度消失的现象，使得训练结果无法继续优化。

（3）tanh：tanh 的数学表达式如下：

$$\tanh(x)=\frac{1-e^{-2x}}{1+e^{-2x}} \tag{2-6}$$

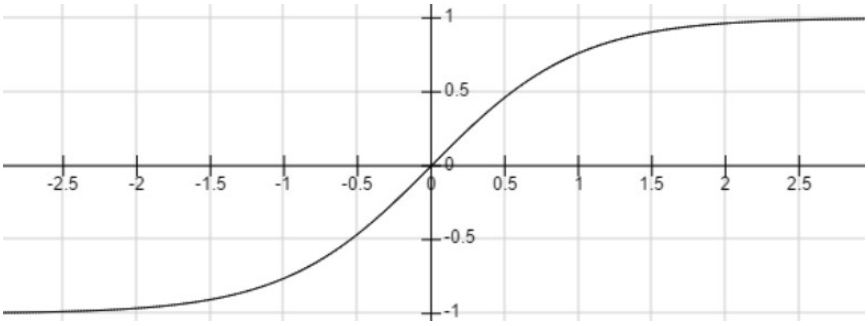


图 2.5 tanh 激活函数的函数曲线

tanh 函数值域为[-1,1]闭区间，同样没有解决 Sigmoid 易出现的梯度消失问题是一个缺憾。不过相对于 Sigmoid 激活函数，tanh 拥有更快的收敛速度，这也是它受欢迎的一个原因。

对于神经网络中，从数据输入的第一层开始，每一层的输出即为下一层的输入，直到最后一层输出层。

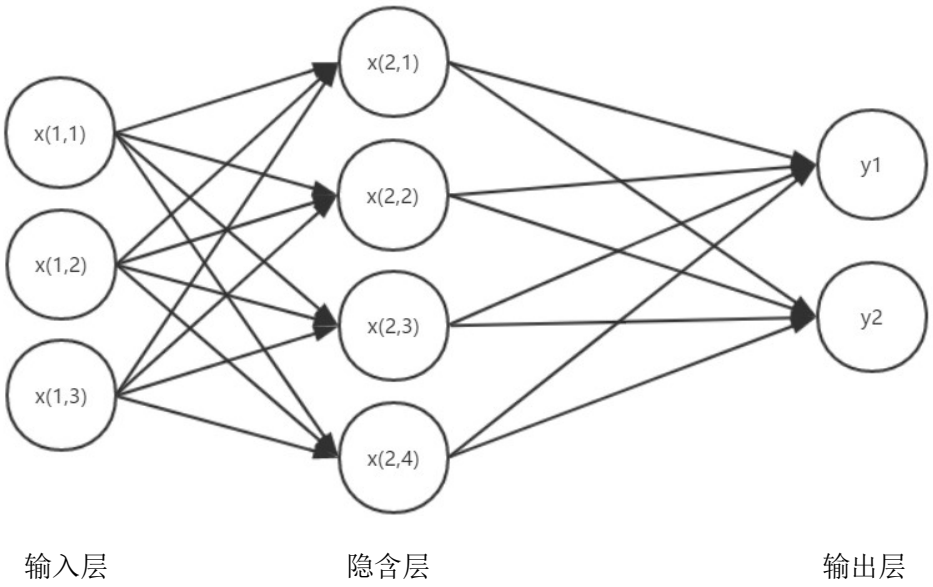


图 2.6 神经网络结构

如图 2.6 即为一个全链接网络。如上图，该网络一共有输入层，隐含层以及输出层三个网络层。对于每一个节点，均存在与上一层所有节点的连接，同时也存在与下一层所有节点的连接。对于每一层连接，由节点$x_{i,j}$连接到节点$x_{i+1,k}$的权重记为$w_{i,k}$，我们另有偏置项 b，这里我们使用 ReLU 函数作为激活函数对隐含层中的结果进行激活操作，那么对于节点 x(2,1)，我们有

$$\begin{aligned} x(2,1) &= \mathrm{ReLU}(\overrightarrow{W}^{T} \bullet \overrightarrow{x_1} + b_1) \\ &= \mathrm{ReLU}(w_{1,1} \bullet x_{1,1} + w_{2,1} \bullet x_{1,2} + w_{3,1} \bullet x_{1,3} + b_1) \end{aligned} \tag{2-7}$$

同样还有

$$\begin{aligned} x(2,2) &= \mathrm{ReLU}(\overrightarrow{W}^{T} \bullet \overrightarrow{x_1} + b) \\ &= \mathrm{ReLU}(w_{1,2} \bullet x_{1,1} + w_{2,2} \bullet x_{1,2} + w_{3,2} \bullet x_{1,3} + b_2) \end{aligned} \tag{2-8}$$

那么我们可以令

$$\overrightarrow{x_1} = \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \end{bmatrix} \tag{2-9}$$

$$\overrightarrow{w_1} = [w_{1,1} \quad w_{2,1} \quad w_{3,1}] \tag{2-10}$$

$$\overrightarrow{w_2}=[w_{2,1}\quad w_{2,2}\quad w_{2,3}] \tag{2-11}$$

$$\overrightarrow{w_3}=[w_{3,1}\quad w_{3,2}\quad w_{3,3}] \tag{2-12}$$

$$\overrightarrow{w_4}=[w_{1,4}\quad w_{2,4}\quad w_{3,4}] \tag{2-13}$$

$$\vec{b}=\begin{bmatrix}b_1\\b_2\\b_3\\b_4\end{bmatrix} \tag{2-14}$$

$$f=\mathrm{ReLU} \tag{2-15}$$

那么，我们就有

$$W=\begin{bmatrix}\overrightarrow{w_1}\\\overrightarrow{w_2}\\\overrightarrow{w_3}\\\overrightarrow{w_4}\end{bmatrix}=\begin{bmatrix}w_{1,1}&w_{2,1}&w_{3,1}\\w_{1,2}&w_{2,2}&w_{3,2}\\w_{1,3}&w_{2,3}&w_{3,3}\\w_{1,4}&w_{2,4}&w_{3,4}\end{bmatrix} \tag{2-16}$$

$$\overrightarrow{x_2}=\begin{bmatrix}x(2,1)\\x(2,2)\\x(2,3)\\x(2,4)\end{bmatrix}=\begin{bmatrix}f(\overrightarrow{w_1}\bullet\overrightarrow{x_1}+b_1)\\f(\overrightarrow{w_2}\bullet\overrightarrow{x_2}+b_2)\\f(\overrightarrow{w_3}\bullet\overrightarrow{x_3}+b_3)\\f(\overrightarrow{w_4}\bullet\overrightarrow{x_4}+b_4)\end{bmatrix} \tag{2-17}$$

即得：

$$\overrightarrow{x_2}=f(W\bullet\overrightarrow{x_2}+\vec{b}) \tag{2-18}$$

以上为一个深度神经网络的示例。**深度**是区分深度学习、传统及其学习的一个关键点。我们已知传统机器学习方法不包含或者仅包含一个隐含层。深度神经网络指的是隐含层数量大于等于 1 的神经网络。该网络系统可以被称为深度学习系统。

在深度神经网络中，每一层网络利用上一层网络的输出来学习识别特征。网络层次越深，整合、重组的特征便越多，且越复杂，网络所能够识别的特征越多，

处理大规模高维度数据更具优势，但同时也会带来更多的计算任务。

2.3 卷积神经网络

以上说明是的是一个全连接神经网络。在本文所述方法中，我们使用了卷积神经网络。当下，AlphaGo、GoogleNet 以及 ResNet 等框架，均用到了卷积神经网络。卷积网络拥有**局部连接**、**权值共享**以及**下连接**的特征。相对于全连接神经网络，卷积神经网络更适合对图像、语音等方面的处理，主要是有以下几个原因：

（1） 图像中包含较多的位置信息。基于图像的特点，每一个像素与四周临近像素会拥有较大的关联，而对于距离较远的像素，往往没有关联较为紧密的关系。全连接网络对于每一个像素，都会平等看待，这在一定程度上不符合我们的预期。而卷积网络拥有的**局部连接**的特征，相对于全连接网络，能够更好地利用像素之间的位置关系，提高信息的利用效率。

（2） 图像处理需要较多的节点数量。比如一个 512×512 像素的图像，输入层节点数量为 512×512=262144 个结点，我们假设其下一层网络有 100 个节点，那么所需要的参数数量为 262144×100=26214400，多余两千万个参数。若图像尺寸扩大或者隐含层节点数量增加，在本文所述需求下，输出节点数与输入节点数相同，那么所带来的计算压力以及内存占用需求将非常庞大。卷积网络拥有**局部连接**的特点，无需连接上一层每一个节点，这样就减少了大量的参数。同时，卷积网络的**权值共享**，对于一组连接共同使用一个参数，再一次减少了参数的数量。

（3） 图像的特征信息数量较多。而我们使用全连接网络增加网络层的数量来提高网络对特征信息的提取，当网络层次数量越多，训练将越加困难，因此，很深的全连接神经网络对于我们的使用并不友好。而在卷积网络中，卷积网络的**下采样**可以在一定程度上提取信息，并减少每层的样本数量，同时，能够减少参数数量，并且在一定程度上提升了其鲁棒性，并提高了性能。

接下来我们说明一下卷积神经网络的工作原理及工作过程。

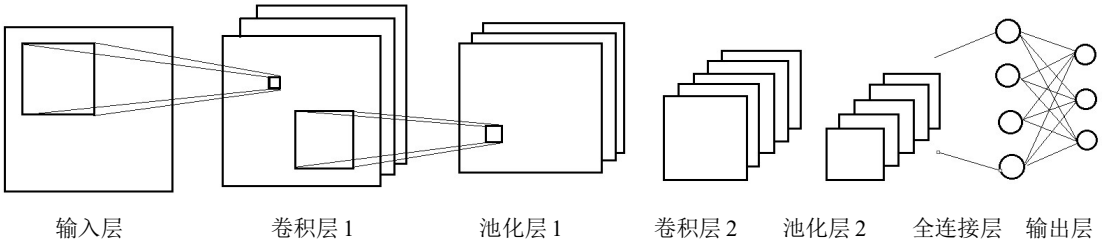


图 2.7 卷积神经网络结构

图 2.7 是一个常见的卷积神经网络模型。通过图 2.7 和图 2.6 的对比，我们能够发现，相对于全连接网络相邻两层网络所有节点全部连接的结构，卷积神经网络有了很大变化。对于每一层网络，全连接网络每层的节点排成一列，而卷积网络，会充分利用位置信息。图 2.7 所示网络，输入层的长和宽分别对应图像的长和宽，在卷积层 1，我们使用多个滤波器(Filter)对输入层进行卷机操作得到多个特征图(Feature Map)。我们在卷积操作以后，又经过池化层的**下采样**处理，得到同等数量的特征图(Feature Map)，再次经过卷积处理后，进入池化层进行池化处理，经过全连接层，得到这个网络的最终结果。

2.4 卷积神经网络的训练

反向传播算法从诞生到如今,一直是训练神经网络以及卷积神经网络的常用方法。可以分为以下几个步骤来描述反向传播算法：

（1） 前向计算神经网络每一层每一个节点（即神经元）的值 x；

（2） 反向计算计算网络中每一个节点的误差项。记损失函数为 E，节点的加权输入为 p，那么误差项 $e=\frac{\partial E}{\partial p}$ ；

（3） 对于每一个节点计算其连接权重 w 的梯度，对于从 i 节点到 j 节点的连接，数学表达式为：$\frac{\partial E}{\partial w}=x\bullet e$，其中，x 为节点 i 的输出，e 为节点 j 的误差项。

（4） 我们更新权重的方式使用梯度下降法(Gradient descen)。对于（3）中所述权重 w，我们有：

$$\triangle w=-\eta\frac{\partial E}{\partial w}=-\eta\bullet x\bullet e \tag{2-19}$$

然而，由于卷积网络的局部连接、下采样操作，以上所述误差项 e 的计算并

不适用于卷积网络。同样，由于卷积网络权值共享，以上所述的权值 w 的更新方法并不适用于卷积网络。那么，对于卷积网络，当 Filter 的数量为 N 时（此时，输出层的深度亦为 N），对于第 i 层的节点，第 i-1 层的每一个输出值 p 均会对第 i 层所有 Feature map 产生影响，故这里不再使用偏导，而使用全导。我们使用 sensitivity map 来记录该结果。若输入层深度为 D，数学表达式如下：

$$e^{i-1}=\sum_{k=0}^{D}e_k^i \bullet W_k^{\mathrm{i}} \bullet f'(p^{i-1}) \tag{2-20}$$

而对于$\frac{\partial E}{\partial w}$的计算，数学表达式为：

$$\frac{\partial E}{\partial w_{i,j}}=\sum_m\sum_n e_{m,n} \bullet p_{i+m.j+n}^{i-1} \tag{2-21}$$

我们同时可以采用批量归一化操作加快我们的训练过程。对于批量归一化操作，我们可以利用以下几个数学表达式来表达：

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m}x_i \tag{2-22}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2 \tag{2-23}$$

$$\widehat{x_i} \leftarrow \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\varepsilon}} \tag{2-24}$$

$$y_i \leftarrow \gamma\widehat{x_i}+\beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{2-25}$$

批量归一化操作可以解决一些训练时收敛速度很慢、梯度爆炸无法继续训练的情况。对于一般的神经网络，批量归一化仍然能够加快训练速度、提高训练精度。

2.5 卷积神经网络应对图像处理

在本文中，我们将图像的去噪问题视为一种辨别学习问题。与以往的去噪方法不同,本文并不是首先去判别噪声模型,而是使用卷积神经网络(CNN)对包含噪声的图像进行处理，将噪声信息与图像原始信息分离。相对于传统方法，使用 CNN 有以下三个优点：首先，CNN 的深层网络结构可以提高发掘图像的特征信息的能力而且拥有较高的灵活度。其次，对于 CNN 的训练，目前已经出现了很

多可以提升其训练效果以及加快训练速度的方法，比如归一化、线性滤波器（Rectifier Linear Unit，ReLU），这些方法均可在一定程度上提高网络的去噪性能。最后，深度卷积神经网络中包含大量的可并行运算，非常适合利用 GPU 来提高运算性能。

本文所使用的方法与以往的神经网络略有不同。对于输入的包含噪声的图像 Y，我们并不去预测干净的图像 X，而是去预测隐藏的噪声 N。我们使用原始图像与包含噪声的图像的差异去训练该神经网络，以加快训练速度以及最终去噪效果。不同于典型的卷积网络模型，我们针对图像去噪设计使用了自己的神经网络。对于输入任意尺寸的图像，该神经网络将输出相同尺寸的噪声，该噪声为图像中所包含的噪声。由式(1-2)我们即可预测不包含噪声的原始图像信息。

我们首先针对特定噪声去训练该神经网络，对比当前较为优秀的去噪算法如 1.4 章节中所述 BM3D、LSSC 以及 TNRD 算法，我们能获得效果相当甚至更优的去噪效果，借助当前强大的 GPU 并行加速，使用 CNN 去噪还能够获得相对较高的性能。

技术的进步，使得大数据、机器学习等方面获得巨大的进步。大规模数据集的使用更加便捷，深度学习的价值进一步体现，卷积神经网络更是在图像处理等方面获得明显的进步，深度神经网应用于图像处理更加便捷。

第三章 深度卷积网络模型

本章节中，将介绍本文所提出的 CNN 模型，来进行图像的去噪任务，并扩展应用它去处理图像去噪的任务。一般来说，训练一个神经网络来达到我们预期的效果需要两个步骤：

（1）网络结构设计

（2）使用训练数据对网络模型进行训练。

对于（1）网络结构设计，本文将介绍一个卷积神经网络结构，并对其进行修改以适合图像的降噪处理。对于（2）神经网络的学习，本文利用包含噪声图像与清洁的图像的差值，对该神经网络进行训练。最后扩展该降噪网络并使其适用于更多情况。

3.1 卷积神经网络基础

在本项目开题之处，为了深入理解深度卷积网络，我们做了以下工作：分别使用 caffe 开源框架以及 Google 与 2015 年末推出的 Tensorflow 框架，构建深度卷积神经网络进行 USTC 验证码的识别。对于获取到的每一个验证码

该验证码图像尺寸为 80×20，我们将其从中分离为尺寸为 20×20 的四个部分，将其分类后进入训练网络。已知验证码中可能出现的字符为除去数字 0、数字 1、字母 I 以及字母 O 的所有大写字母以及阿拉伯数字共 32 个字符。输入数据训练为 n 个尺寸 20×20 的图像，我们将其转化为单通道图像并将其转化为 n×20×20×1 的四维数据，训练的 label 为长度为 32 的独热码。我们使用最小二乘法则来计算损失函数 loss 值，即

$$loss = (pred - label)^2 \tag{3-1}$$

使用最速下降法，并使用 adam 优化算法进行训练。

那么，该数据会进入该神经网络的训练：

（1） 第一层卷积层:在这一次卷积网络中使用 VALID 填充类型，VALID 类型不会在原有数据基础上填充数值。这里设置步长为[1,1]，我们将每一个卷积核的尺寸设定为 5×5，我们共使用 20 个卷积核分别对网络

进行卷机操作，在这里，获得 20 个特征图。一般来说，对于不同的特征图，与之对应的是不同的特征。对应的，我们设置了 20 个偏置量，并使用 ReLU 激活函数来完成这一层网络。

（2） 第一层池化层：在这一层池化中，池化窗口设定为 2×2，池化操作为 max pooling，即在每一个 2×2 窗口中，最大值为输出值，同时，也减少了数据的规模。该层输出尺寸为 5×5×20.

（3） 第二层卷积层：在该层卷积层中，同样使用了 VALID 填充类型。第二层中，我们将卷积核的尺寸设置为 5*5，对于以上输出结果，我们使用 50 个这样的卷积核对其进行卷机操作，对应的，设置了 50 个偏置量，对于以上结果，我们使用 ReLU 激活函数进行激活操作，最终该层输出 50 个特征图（feature map）。

（4） 第二次池化层：继续使用 2×2 的窗口对上层的输出进行 max pooling 池化，该层的输出尺寸为 2×2×50。

（5） 全连接层：该层拥有 500 个节点（即神经元）。对于上层的输出，我们首先改变其尺寸，将其 reshape 成 2×2×50=200 单位长度的一维向量，设置偏置量，对于最后的运行结果，我们对其进行 ReLU 激活。

（6） 添加 Dropout 操作来减少神经网络中常常出现的过拟合现象。

（7） 输出层使用全连接的方式来实现，该层节点数对应分类数量。该识别中共有 32 种输出，故该层拥有 32 个节点（即神经元），我们对以上过程的输出使用 softmax 激活函数对其进行激活操作。

以上为该神经网络的结构。由于该数据集特征明显，在充分训练后，得到无限接近 100%的正确率。

3.2 深度卷积网络应用于图像降噪

由 3.1 已经描述了一个常见的深度卷积神经网络，考虑到图像去噪的特征，输入的包含噪声的原图像应与输出噪声图像拥有相同的尺寸。那么对于尺寸为 256×256 的图像，若取得最终理想的输出，那么最终的全连接层所需参数数量至少为(256×256)×(256×256)= 4,294,967,296，如此庞大的数量必然会带来巨大的性能的下降，且无法针对图像的多种尺寸进行灵活的改变。于是，在该卷积神经网络中，删除了全连接层。为了避免信息的丢失，本文所述方法删除了全部的池

化层，全部使用卷积层、批量归一化（Batch Normalize）以及激活层处理。

那么，对于输入数据，首先对数千张图像进行预处理。便于训练，我们将所有图像压缩为尺寸为 256×256 的缩略图，将其转为单通道灰度图。手动添加高斯噪声后作为训练数据，噪声为期望网络输出的结果，并保存为 mat 格式。

对 3.1 所述网络进行修改，网络结构如下：

（1）输入：首先将输入的尺寸为 256×256 的灰度图 reshape 为尺寸为 256×256×1 的数据格式，并做归一化操作，使数据集所有数据的范围控制在 0 至 1 之间。

（2）第一层卷积层：在这一层网络，同样的，我们将每一个卷积核的尺寸设定为 5×5，我们共使用 24 个卷积核分别对网络进行卷机操作，填充模式为 SAME，添加偏置项，进行批量归一化操作（Batch Normalize）后使用 ReLU 激活函数进行激活操作，该层输出结果为 24 个 feature map。

（3）第二层卷积层中，同样使用了尺寸为 5×5 的 24 个卷积核对上一层网络的输出进行卷积操作，填充模式同样为 SAME，添加偏置项、并进行批量归一化操作（Batch Normalize）后，进行 ReLU 激活操作，该层的输出结果为同样为 24 个 feature map。

（4）第三次卷积网络中，继续使用 5×5 的尺寸对上一层网络的输出结果进行卷积操作，卷积核的数量仍然为 24，SAME 填充模式、偏置项、批量归一化以及 ReLU 激活函数处理后，得到 24 个 feature map。

（5）最后一层卷积层，亦为该网络的输出层。在该层网络中，对应单通道灰度图，仅使用一个卷积核处理，该卷积核尺寸设定为 5×5，填充类型为 SAME，以保证输出尺寸与输入尺寸相同。该层输出为本网络的输出。

代码描述：见附录【1】

本文所述方法中，对于 Y=X+N 的加性高斯噪声模型，该网络的输出为 y≈Y-X=N，对于包含噪声的输入图像，我们使用 X=Y-y 来获得预测的清洁图像。于是损失函数的计算，采用最小二乘方法：

$$loss = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}(y_{i,j} - (Y_{i,j} - X_{i,j}))^2 \tag{3-2}$$

式中 m、n 分别为图像的宽度以及高度。

对于固定尺寸的图像，可以使用以下表达式来代替：

$$loss = \sum_{i=1}^{n}\sum_{j=1}^{m}(y_{i,j} - (Y_{i,j} - X_{i,j}))^2 \tag{3-3}$$

代码描述见附录【3】

综上，假定网络的深度为 d，本文所述卷积神经网络的结构可分为以下三种类型:

（1） 对于第一层网络，使用卷积操作并使用 ReLU 激活。卷积核尺寸为 5×5×c，卷积核的数量为 24 个，其中 c 为通道数。对于单通道灰度图，c=1，对于彩色图像，c=3 或者 c=4（RGB、HSL 等情况下 c=3，RGBA 等情况下 c=4）。

（2） 对于第二层到第 d-1 层网络，使用卷积操作后，进行批量归一化（Batch Normalize）处理并使用 ReLU 激活。卷积操作的卷积核尺寸为 5×5×24。

（3） 对于最后一层网络，对于上网络的输出进行卷积操作，我们分别使用 c 个卷积核进行，卷积核尺寸为 5×5×24。该层网路的输出即为本网络的输出。

总之，该去噪方法有以下几个特点：

（1） 利用噪声进行学习，并借助批量归一化（Batch Normalize）等方法对训练过程进行加速。

（2） 该算法拥有较高的并行性，借助 GPU 加速，可以获得较高性能。

（3） 网络的训练是图对图的，不需要过多的调参。神经网络中的隐含层通过充分的学习可以对噪声信息进行有效的区分。

图像的去噪往往要求输入尺寸与输出尺寸完全相同，这对于传统去噪方法较容易出现边缘去噪不净或者边缘画质损失等情况，TNRD 去噪方法会在算法的每一个步骤前进行像素填充以实现输出尺寸与输入尺寸相同。而在本文所述方法中，对于图像的边缘，首先填充 0，再进行卷积操作，这样我们便可以保证输出图像将与输入图像具有相同的尺寸。最终结果证明，即使是简单的 0 填充，没有对输出结果产生不利影响　　。

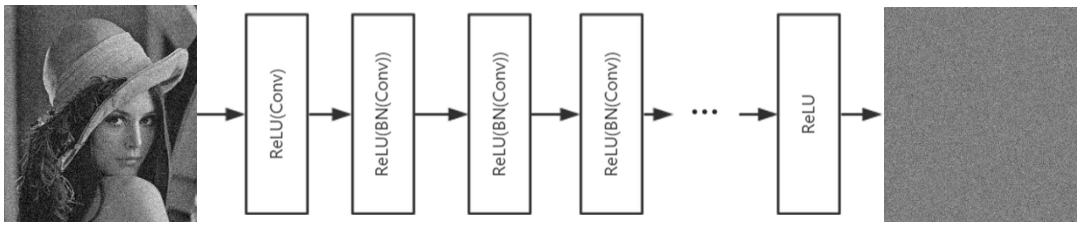


图 3.1 本文所提出的去噪网络的结构

首先，我们将使用两种方法来衡量算法的去噪效果的优劣，信噪比（SNR）以及峰值信噪比（PSNR）。首先假设图像尺寸为 m×n，且该图象为单通道灰度图。在这里，我们有两个这样的图：p 和 p_r，其中 p 为对噪声去噪处理后得到的图像，p_r 是不包含噪声的原图。那么计算信噪比（SNR）的方式如下：

$$SNR = 10 \bullet \log_{10}(\frac{p\_r}{p\_r - p})^2 = 20 \bullet \log_{10}(\frac{p\_r}{p\_r - p}) \tag{3-4}$$

那么我们首先定义他们的均方差：

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} || p_{i,j} - p\_r_{i,j} ||^2 \tag{3-5}$$

那么我们可以由以下公式获得 PSNR 值：

$$PSNR = 10 \bullet \log_{10}(\frac{MAX^2}{MSE}) = 20 \bullet \log_{10}(\frac{MAX}{\sqrt{MSE}}) \tag{3-6}$$

其中，MAX 为当前情况下允许的最大值。对于单色 8bit 图像，MAX=2^8-1=255。

同时，在该卷积网络中，批量归一化发挥了重要的作用，如图 2.2 所示，在本数据集测试条件下，首先固定卷积网络中基于梯度的优化算法为 Adam，且使用同样的网络结构及相同的网络参数。相较于未使用批量归一化的情况下，使用批量归一化后，在训练初期略微落后，而后训练速度明显快于未使用批量归一化的情况，而且训练结果更优。

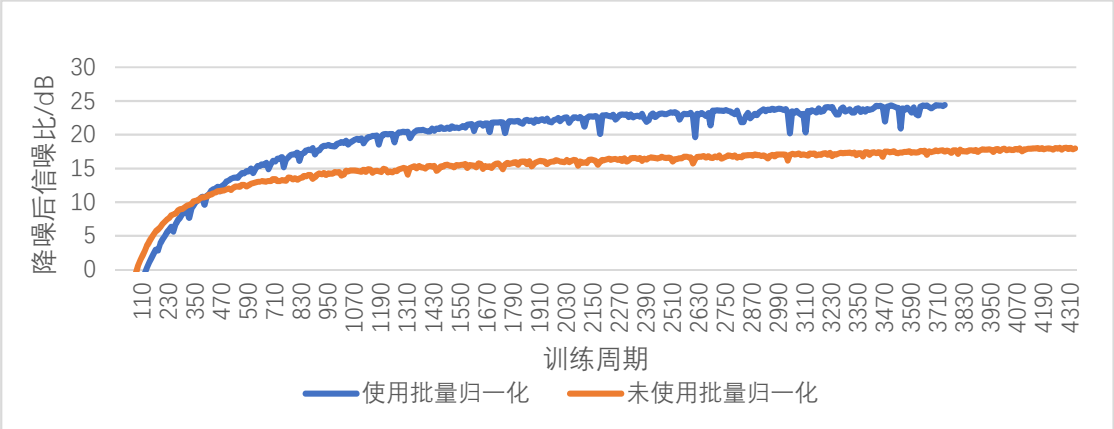


图 3.2 是否使用归一化带来影响对比

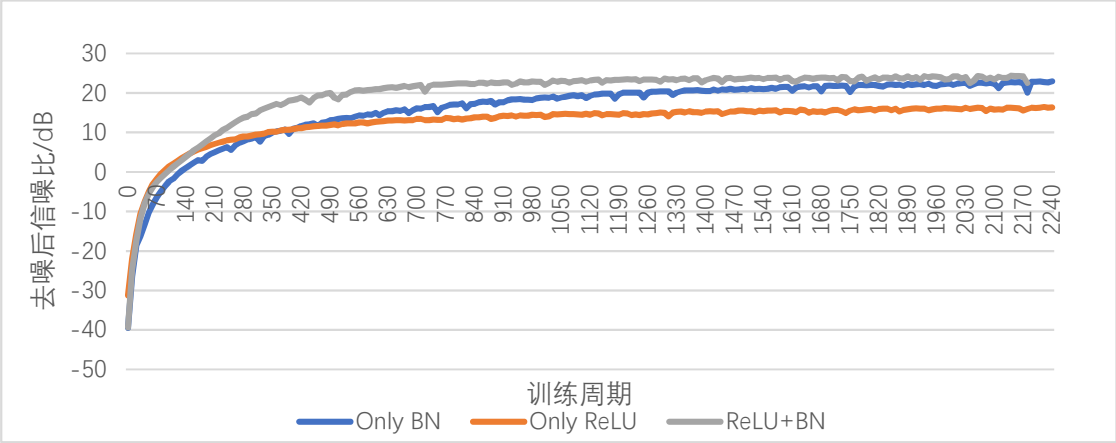


图 3.3 三种网络训练效果对比

如图 3.2 所示为在使用相同的网络结构，相同的训练参数，以及相同的梯度下降优化算法（Adam）下，训练后获得的信噪比 PSNR 的对比。图中我们可以看到，没有批量归一化的情况下，收敛速度在 500 次循环后变得十分缓慢，使得网络训练非常艰难。在没有 ReLU 激活操作的情况下，神经网络的训练速度同样不理想。当该卷积网络结合了 ReLU 激活函数的激活处理以及批量归一化处理后，它的收敛速度更快，且更稳定。

3.3 功能性拓展

以上所述卷积神经网络，仅针对特定特定尺寸且具有特定高斯噪声的图像进行噪声的区分。接下来，将对其进行拓展。

A．首先考虑尺寸问题。由 3.2 所述，神经网络每一层均由卷积操作、批量归一化操作、以及激活操作构成，并不包含全连接层。其中，批量归一化以及激活操作均对数据的尺寸没有要求。由 2.3 中卷积操作的原理，卷积操作对图像的尺

寸没有固定的要求。且该网络不包含对尺寸敏感的全连接。这就意味着，对于同样噪声类型的图像，我们不需要重新训练不同的尺寸的网络亦可进行对不同尺寸图像的去噪处理，且避免使用图像切割等方法。

对此，修改网络详见附录【2】

这里的输入层尺寸并不是固定的，其尺寸根据图像尺寸自适应。

其中，im.size 为输入图像的尺寸，是一个二元数组。最终，实现了对不同尺寸图像的去噪处理，而不仅仅针对 256×256

图 3.4 对于更大尺寸图像进行去噪

图 3.4 中，使用已有网络以及经过训练的模型对尺寸为 512×512 的图像进行去噪处理，该高斯噪声 $\sigma=16$，去噪效果与 256×256 图像的去噪效果相当，去噪 PSNR 为 32.79dB。

B．目前已进行处理的图像均为单通道灰度图。由 3.2 中网络结构，针对彩色图像不同的通道数，我们选择 c=3 或者 c=4，即可进行对彩色图像的去噪。

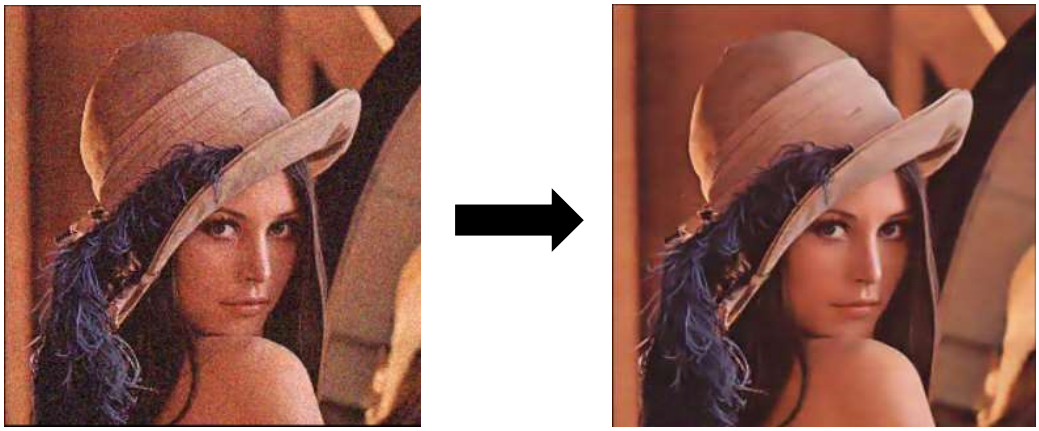


图 3.5 对彩色图进行去噪处理

图 3.5 中，对彩色的图进行去噪处理中，算法也表现出不错的效果，该高斯噪声 $\sigma = 16$，最终输出图像 PSNR 为 32.97dB。

C．目前较为流行的去噪算法进行去噪处理前多需手动指定噪声水平以针对特定的噪声进行去噪处理，或者先对噪声水平进行估计，再使用相应的参数进行去噪。最终的去噪效果不仅和去噪算法有关，也和估计噪声水平的精度、准确度有关，且对于非高斯噪声，去噪算法往往没有非常好的表现。对此，对该去噪网络再一次进行拓展。我们使用 $\sigma = 6 \sim 25$ 的噪声对网络进行训练，以使网络能够对不同噪声进行去噪。实验结果表明，该方法能够获得不错的成果。

第四章 实验结果及分析

4.1 训练数据

在本章节中，讲述运行环境、训练数据、训练结果、性能对比、结果对比、包括 针对特定噪声 针对随机噪声（需要 BM3D 使用多重去噪标准进行，打一个表格，进行对比 BM3D 对不不同标准 以及使用本文所述的方法，另外附一张对比图表）

训练数据：对于训练数据以及测试数据，首先对 2249 张图片进行缩放处理，制作 256×256 的缩略图，并以此作为训练数据以及测试数据的原始数据。

针对固定噪声水平的高斯噪声的去噪，我们对原始数据进行处理，人工添加 $\sigma=16$ 的高斯噪声，并使用该高斯噪声作为训练的标签。

对于不固定噪声水平的高斯噪声的去噪，同样使用已经生成的 2249 张图像进行处理。对于 2249 张图像，手动随机添加 $\sigma\in[6,25]$ 的高斯噪声，并使用该噪声作为训练标签，进行训练。在此环节中，将涉及到更多的特征，网络深度有所增加。

4.2 测试数据及对比

在固定噪声水平的高斯去噪测试中，在灰度图部分，使用经典的 camearman、boat、lena、house、barbara、pappers、couple、hill 以及 man 进行测试。在彩色图部分，使用经典的 lena 以及 flower 图进行测试。对比的其他算法为传统的平均滤波算法以及 BM3D 算法。参与对比的项目有去噪后的峰值信噪比（PSNR）、算法运行时间以及具体细节对比。



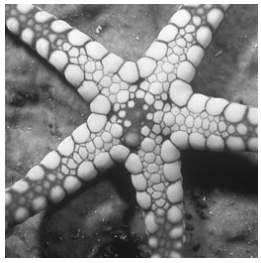



图 4.1 灰度图部分测试图

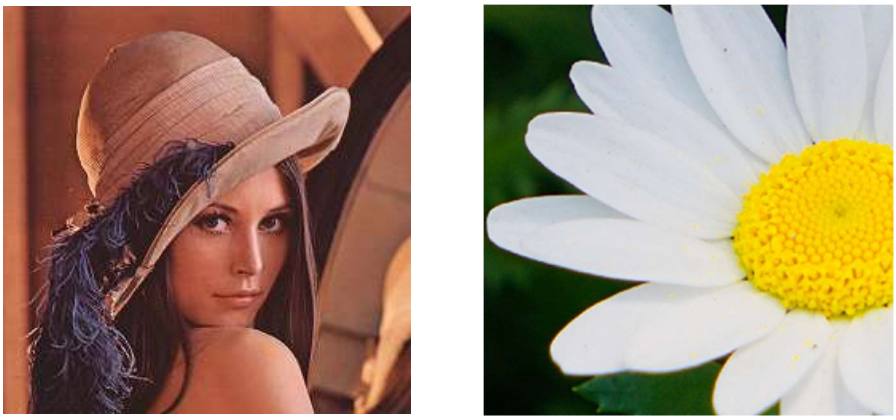


图 4.1 彩色图测试部分

在不固定噪声水平的高斯噪声去噪测试中，使用经典的 lena 图像进行。对于 lena 图像，已经实现做好了添加了 $\sigma \in [6,25]$ 的高斯噪声的测试样例，方便进行测试的进行。这里参与对比是 BM3D 算法在指定 $\sigma = 10$ 以及 $\sigma = 20$ 情况下对以上测试样例进行去噪。

对于以上所提出的训练以及测试任务，均使用 PyCharm 2017.1.2 进行。所使用 PC 环境为 CPU：Intel(R) Core(TM) i7-4710HQ 2.5GHz，GPU：Nvidia GTX850M。进行测试前，对固定噪声水平的去噪模型进行了 9 个小时的训练，对不固定噪声水平额去噪模型进行了 12 个小时的训练。

4.3 测试结果及分析

首先针对固定噪声水平的高斯去噪。在这次对比中，分别对本文提出的 CNN

去噪、BM3D 去噪、以及传统的平均滤波去噪分分别进行去噪后峰值信噪比（PSNR）以及运行时间的对比。该测试在高斯噪声水平为$\sigma \in 16$情况下进行。

a.包含噪声的图像　　b.去噪后图像　　c.提取出来的噪声

图 4.2 使用 CNN 去噪

图 4.2 所示为该去噪模型去噪过程的展示，a 图即为 b 图与 c 图的叠加。该去噪网络从中提取出了 c，即得到了 b。图 4.3 展示了该去噪网络的细节保留能力。通常情况下，去噪会丢失部分细节，仔细观察女孩衣服上的条纹以及 lena 的头发，即可发现，本文所述网络中，细节并没有过多的丢失。

图 4.3 细节对比 左：原图 右：去噪后的图像

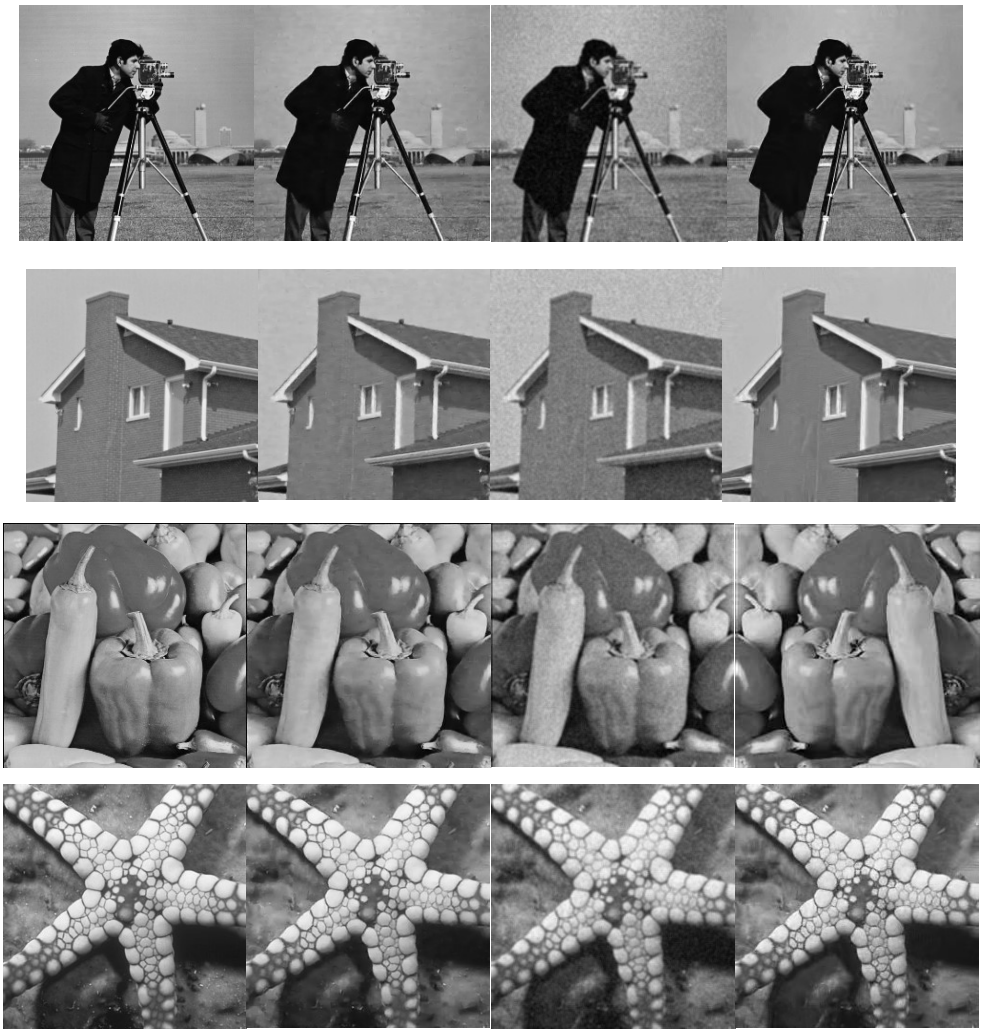


图 4.4 包含噪声原图、使用 CNN 去噪、BM3D 去噪、平均滤波去噪对比。
从左到右分别为包含噪声原图、使用 CNN 去噪、BM3D 去噪、平均滤波去噪

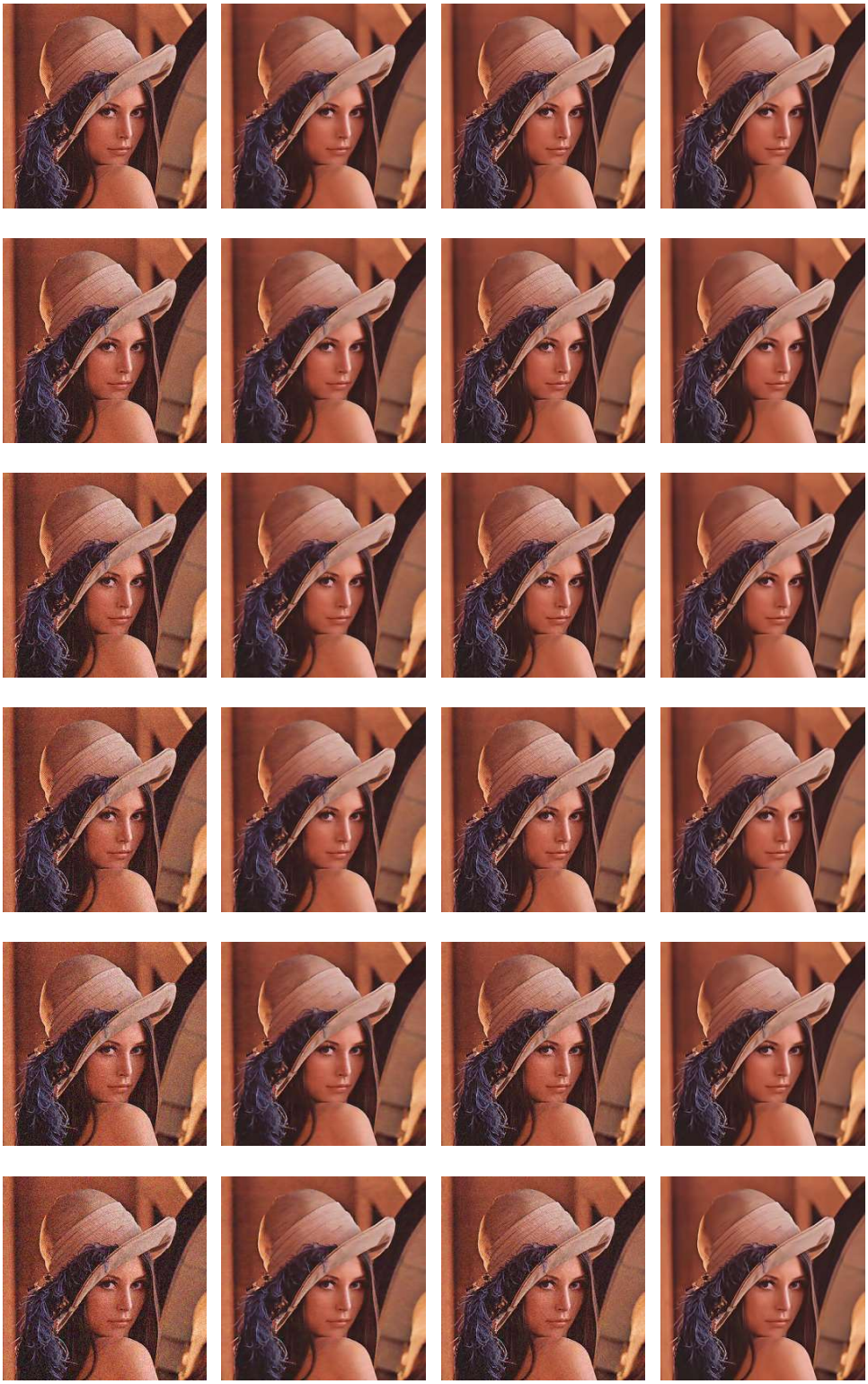
表 4.1 为展示了算法最终的去噪效果。本测试中固定高斯噪声水平$\sigma=16$。首先，由于本文所提出算法同时追求去噪效果以及高性能，使用网络较为简单。在这种较为简单的网路下，仍能够具有与 BM3D 算法相当的去噪性能，实验结果展现了卷积神经网络的巨大潜力。图 4.2 展示了几种算法的运行时间对比。得益于强大的 GPU 并行运算性能，本文提出卷积网络算法对于 BM3D 算法拥有了几十倍的性能提升。而该对比分别基于 CPU 以及 GPU，这样的对比难以体现出算法的高性能。故我们又指定使用 CPU 进行运算，结果仍然表现出了不俗的性能。考虑到该算法能够获得与 BM3D 相近的去噪性能，可见，实验结果是令人欣慰的。

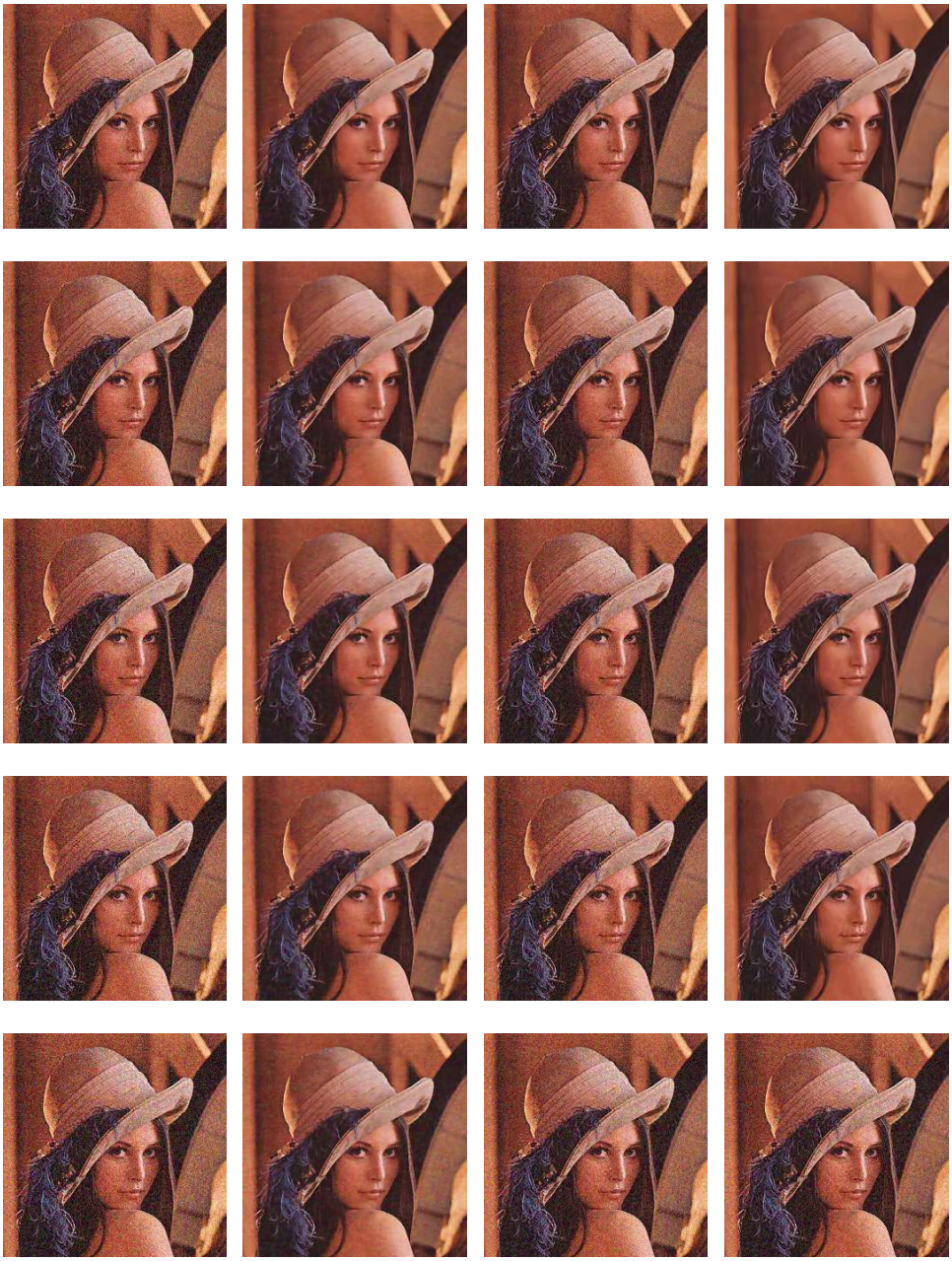
PSNR/dB	CNN	NR_mean	BM3D
01.png	31.2471955	25.27015454	**31.38458131**
02.png	33.28404097	29.90750673	**34.36705262**
03.png	31.73267806	26.13832255	**32.11005128**
04.png	**31.0674927**	27.51529365	30.55176983
05.png	**31.5934707**	26.90759667	31.22319071
06.png	**30.77772792**	25.18810662	30.62883149
07.png	**31.09174824**	25.70649449	30.95222418
08.png	**35.25528124**	30.6053586	33.8037007
09.png	30.10060597	24.6950151	**32.35011168**
10.png	31.45166937	28.11878812	**31.74215176**
11.png	31.52627494	28.90883703	**31.58416244**
12.png	31.27141109	28.01088329	**31.62153288**

表 4.1 去噪最终峰值信噪比（PSNR）对比

时间/s	CNN	CNN(CPU)	NR_mean	BM3D
01.png	**0.201**	0.231	2.135	14.96
02.png	**0.199**	0.236	2.094	15.49
03.png	**0.200**	0.231	2.079	15.09
04.png	**0.202**	0.234	2.200	14.49
05.png	**0.203**	0.226	2.169	14.69
06.png	**0.202**	0.226	2.116	14.38
07.png	**0.196**	0.231	2.519	14.48
08.png	**0.446**	0.88	8.833	31.15
09.png	**0.449**	0.961	9.006	29.75
10.png	**0.447**	0.896	9.030	29.04
11.png	**0.448**	0.875	8.907	35.26
12.png	**0.442**	0.882	8.569	30.84

表 4.2 去噪算法运行时间对比





包含噪声图　　CNN　　BM3D-10　　BM3D-20

图 4.5 对于不同噪声水平图像的去噪

从上到下噪声水平分别是 $\sigma = 4, 6, 8, ..., 26$，从左到右分别为包含噪声的图像、使用该神经网络去噪后的图像、BM3D 算法设定 $\sigma = 10$ 去噪后的图像以及 BM3D 算法设定 $\sigma = 20$ 去噪后的图像。

图 4.5 展示了去噪网络以及 BM3D 算法对不同噪声情况下的去噪效果。有图可见，当 BM3D 算法在噪声水平设定准确时，去噪效果较好。当噪声水平大于设定噪声水平时，会出现较多的噪声残留现象。当噪声水平小于设定噪声水平时，则容易出现细节丢失的问题。而对于使用不同噪声进行过训练的去噪神经网络，

结果表现较为稳定，如表 4.3 所示。

PSNR/dB	BM3D-10	BM3D-20	CNN
σ =6	**36.97356015**	33.40870956	33.38923828
σ =8	**36.50051979**	33.34458741	33.31812452
σ =10	**35.7388998**	33.27873705	33.24743425
σ =12	**34.45901039**	33.19661753	33.14199662
σ =14	**32.62281171**	33.09027261	33.06435331
σ =16	30.68511527	**32.85377552**	32.83433966
σ =18	29.04540531	32.73299558	**34.44476086**
σ =20	27.62150262	32.31388431	**34.17393593**
σ =22	26.46328115	31.85535024	**33.85981695**
σ =24	25.53295404	31.13653877	**33.49729805**
σ =26	24.6431172	24.6431172	**33.02179337**

表 4.3 不同噪声水平下去噪效果

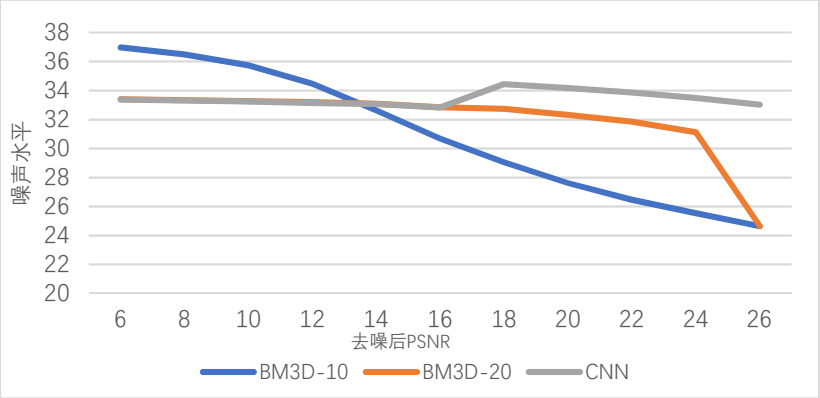


图 4.6 不同噪声水平下去噪效果

图 4.6 可以非常直观的对比集中去噪算法的去噪效果。其中，使用卷积网络进行去噪表现出了较强的稳定性。即使是一个较为简单的网络结构，在噪声水平为 $\sigma = 6$ 到 $\sigma = 26$ 中均表现出了不错的成绩。在该轮测试中，卷积网络同样表现出了较高的性能，这里不再详述。

综上，使用卷积网络进行去噪处理，可以同时获得较好的去噪效果以及优秀的性能表现，结果令人欣慰。

第五章 全文总结及展望

本文所所提出图像去噪方法，不同于传统方法，而是基于深度卷积神经网络，使用噪声对网络进行训练，使得网络可以从中提取出噪声信息，同时还具有极高的运算性能。在训练中，结合了批量归一化等措施以加快网络的训练速度。我们使用不同的训练方式使网络获得对指定噪声水平进行高质量去噪、对未知噪声水平进行有效去噪。同时，得益于它较高的性能，同时可以使用 GPU 进行加速，该方法更适合在移动终端或者服务器端对图像进行处理。深度学习的发展迅速，必将在更多其他领域施展更多的用途。

参考文献

[1] 顾晓东, 郭仕德, 余道衡. 一种基于 PCNN 的图像去噪新方法[J]. 电子与信息学报, 2002, 24(10):1304-1309.

[2] Zhang K, Zuo W, Chen Y, et al. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 2017, PP(99):1-1.

[3] Jain V, Seung H S. Natural image denoising with convolutional networks[J]. Advances in Neural Information Processing Systems, 2010:769-776.

[4] 肖小奎, 黎绍发. 加强边缘保护的 Curvelet 图像去噪方法[J]. 通信学报, 2004, 25(2):9-15.

[5] 陈先昌. 基于卷积神经网络的深度学习算法与应用研究[D]. 浙江工商大学, 2014.

[6] 丁炜. 基于非局部模型与字典学习的自然图像去噪方法研究[D]. 西安电子科技大学, 2013.

[7] 张林刚. 基于 Treelet 变换的图像去噪[D]. 西安电子科技大学, 2012.

[8] Marc Lebrun, An Analysis and Implementation of the BM3D Image Denoising Method, Image Processing On Line, 2 (2012), pp. 175–213. https://doi.org/10.5201/ipol.2012.l-bm3d (accessed 2017/5/17)

[9] hanbingtao，零基础入门深度学习，https://www.zybuluo.com/hanbingtao/note/433855 (accessed 2017/3)

[10] v_JULY_v ， CNN 笔记：通俗理解卷积神经网络，http://blog.csdn.net/v_july_v/article/details/51812459 (accessed 2017/3)

[11] 科言君，CNN(卷积神经网络)、RNN(循环神经网络)、DNN(深度神经网络)的内部网络结构，https://www.zhihu.com/question/34681168 (acesed 2017/4)

[12] 无事扯淡，深度学习笔记(6)卷积，http://www.jianshu.com/p/3209cc6d80ee (accesed 2017/5)

[13] Not_GOD，第六章 深度学习（上），http://www.jianshu.com/p/3716fa796677 (accesed 2017/4)

[14] Google Inc. TensofFlow 官方文档，https://www.tensorflow.org/ (accesed 2017/2)

[15] 姜戈，机器学习简史，http://www.jianshu.com/p/91c00afdc0a1 (accesed 2017/5)

附录

【1】 附代码 1

```
x_image = tf.reshape(x, [-1, 256, 256, 1])

W_conv1 = weight_variable([5, 5, 1, 24])  # 第一层卷积层
b_conv1 = bias_variable([24])  # 第一层卷积层的偏置量
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

W_conv2 = weight_variable([5, 5, 24, 24])  # 第二次卷积层
b_conv2 = bias_variable([24])  # 第二层卷积层的偏置量
h_conv2 = tf.nn.relu(batchnormalize(conv2d(h_conv1, W_conv2) +
b_conv2))

W_conv3 = weight_variable([5, 5, 24, 24])  # 第三次卷积层
b_conv3 = bias_variable([24])  # 第二层卷积层的偏置量
h_conv3 = tf.nn.relu(batchnormalize(conv2d(h_conv2, W_conv3) +
b_conv3))

W_conv4 = weight_variable([5, 5, 24, 1])  # 第四次卷积层
b_conv4 = bias_variable([1])  # 第二层卷积层的偏置量
h_conv4 = conv2d(h_conv3, W_conv4) + b_conv4
y = tf.reshape(h_conv4, [-1, 65536])
```

【2】 附代码 2

```
x_image = tf.reshape(x, [-1, im.size[0], im.size[1], 1])

W_conv1 = weight_variable([5, 5, 1, 24])  # 第一层卷积层
b_conv1 = bias_variable([24])  # 第一层卷积层的偏置量
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

W_conv2 = weight_variable([5, 5, 24, 24])  # 第二次卷积层
b_conv2 = bias_variable([24])  # 第二层卷积层的偏置量
h_conv2 = tf.nn.relu(batchnormalize(conv2d(h_conv1, W_conv2) +
b_conv2))

W_conv3 = weight_variable([5, 5, 24, 24])  # 第三次卷积层
b_conv3 = bias_variable([24])  # 第二层卷积层的偏置量
h_conv3 = tf.nn.relu(batchnormalize(conv2d(h_conv2, W_conv3) +
b_conv3))
```

```
W_conv4 = weight_variable([5, 5, 24, 1])  # 第四次卷积层
b_conv4 = bias_variable([1])  # 第二层卷积层的偏置量
h_conv4 = conv2d(h_conv3, W_conv4) + b_conv4
y = tf.reshape(h_conv4, [1, -1])
```

【3】 附代码 3

```
loss = tf.reduce_sum((y_ - y)××2)
train_step = tf.train.AdamOptimizer(2e-4).minimize(loss)
```