# 实验5《基于Keras卷积神经网络实现》

实验学时： 2 实验地点： 二综204 实验日期： 2019/11/20

1. **问题描述**

掌握基于TensorFlow的高级API框架Keras的基本用法，仍然通过MNIST手写数字体数据集，学会搭建给予Keras API的卷积神经网络，并用来识别手写数字体。

1. **设计简要描述**

**1．准备工作**

**1.1． 导入包和数据集**

首先请导入包，该步骤参考实验一

**1.1.1 安装h5py包**

在本实验是在Ubuntu18.04下完成的，其中包h5py需要自己自行安装。操作步骤如下:

首先Ctrl+Alt+T打开终端，然后输入pip install h5py完成安装

**1.1.2 导入相关包**

本教程需要从Tensorflow导入一些Keras的包。请从keras的models模块中导入Sequential,从layers模块中导入InputLayer,Input,Reshape,MaxPooing2D,Conv2D,Dense,Flatten

**1.2. 载入数据**

请加载MNIST数据集（该数据集大约有12MB，如果给定的地址里没有文件，它将自动下载）

**1.3．配置神经网络**

配置神经网络参数，包括定义图像中每个维度中的像素个数（img\_size）、图像的像素总个数（img\_size\_flat）、用来重塑图像的高度和宽度的元组（img\_shape表示）、用来重塑图像的高度，宽度和深度的元组（img\_shape\_full表示，用于在Keras重塑）、类别数量（num\_classes）、图像通道数（num\_channels）。打印并查看上述定义和赋值的变量

**1.4.****绘制图像的辅助函数**

实现的绘制图像的辅助函数plot\_images并打印测试集前9张图片来检测该函数是否正确

**1.5.绘制错误分类图像的辅助函数**

**2.序列模型**

**2.1模型框架**

Keras API有两种构建神经网络的模式。最简单的是顺序模型，它只允许按顺序添加图层

**2.2模型编译**

神经网络现已定义，必须通过添加损失函数，优化器和性能指标来最终确定。这在Keras中被称为模型“编译”。我们可以使用字符串定义优化器，或者如果我们想要更多地控制其参数，那么我们需要实例化一个对象。 例如，我们可以设置学习率。

请从tensorflow.python.keras.optimizers包中导入Adam，并设置优化器optimizer的学习率为1e-3，然后编译模型，将参数optimizer设置为optimizer， loss设置为'categorical\_crossentropy'，矩阵 metrics设置为'accuracy'

**2.3 训练**

现在模型已经完全定义了损失函数和优化器，我们可以使用序列模型的fit（）函数进行训练，此函数采用numpy数组作为输入。并使用给定的批量大小执行给定数量的训练epoch。 一个epoch是对整个训练集的充分利用。因此，对于10个epoch，我们将在整个训练集上随机迭代10次。

请拟合模型：epochs设置为1，batch\_size设置为128

**2.4评估与性能指标**

现在模型已经过训练，我们可以在测试集上测试它的性能。在这里，也使用numpy数组作为输入。

请用model.evaluate方法来评估其在测试集上的性能，并将结果返回给result，然后查看result的性能指标。值得注意的是，result的返回值可以通过键值对的形式通过zip（key，value）函数访问，其中参数key可传入model的成员变量metric\_name作为键,而参数value可传入result作为值

**2.5 预测**

我们还可以预测新图像的分类。 在这里，我们将只使用测试集中的一部分图像，即测试集的前九张图像。请对比一下预测的类别和真实的类别。提示，可以使用model类的predict（x）方法。该方法返回模型所预测的输入的图像x的类别。获取预测的结果后，请将预测的结果用plot\_images(images,cls\_true,cls\_pred)方法显示出来。

**2.6 错分类的图片**

我们可以从测试集中绘制错误分类图像的一些示例。请从测试集中得到所有图像的预测类，参数返回给y\_pred，然后将预测的类数从One-Hot编码数组转换为整数赋值给cls\_pred，最后调用plot\_example\_errors绘制y\_pred的前九张错分类的图像

**3. 功能模型**

Keras API还可用于使用功能模型构建更复杂的网络。具体做法就是每次调用Keras 创建神经网络的API后都会创建并返回一个可被其他API作为参数调用的实例。这允许我们构建比序列模型所允许的模型更复杂的计算图。

**3.1 模型框架**

Keras配置功能模型介绍

**3.2 模型编译**

我们现在已经使用其输入和输出定义了模型的体系结构。我们现在必须创建一个Keras模型并使用损失函数和优化器对其进行编译，以便为训练做好准备。首先需用以下代码来完成导入model这个包（不要求学员实现）

**3.3 训练**

现在模型已经完全定义了损失函数和优化器，请使用和序列模型相同的fit（）函数进行训练，此函数采用numpy数组作为输入。其epoches=1,batch\_siez = 128

**3.4评估**

现在模型已经过训练，我们可以在测试集上测试它的性能。 这也使用numpy数组作为输入。请参照2.4请用model.evaluate方法来评估其在测试集上的性能，并将结果返回给result，然后查看result的性能指标，result的返回值分别对应的名称是model.metrics\_names

**3.5错分类的图片**

**4. 保存和加载模型**

**4.1保存Keras模型**

首先设置模型的保存路径（可自行设置）。实例代码：

path\_model = 'model.keras'

可以简单的使用model.save（）来保存模型，参数就是上一步的路径变量。请保存上述的模型model2到你自己设置的路径中（假设路径参数就是path\_model）

**4.2删除模型**

Keras可使用model. del（）来删除模型。请用del来删除model2

**4.3加载模型**

我们需要导入如下包来加载模型：

from tensorflow.python.keras.models import load\_model

**4.4用加载的模型来预测测试集的前9张图片**

请用model3来完成测试集前九张图片的预测

**5权重和输出的可视化**

**5.1画卷积权重的辅助函数**

请参考实验3的相应部分来完成绘制卷积权重的辅助函数plot\_conv\_weights(weights, input\_channel=0) ，输入参数weights是待可视化的权重，input\_channel是输入通道的数量，其缺省值为0

**5.2得到层**

Keras有一种简单的方法来列出模型中的图层。即调用summary()方法。请调用summary方法查看model3的所有层的信息，并把输入层赋值给layer\_input，第一层卷积层赋值给layer\_conv1，第二层卷积层辅助给layer\_conv2（提示:summary返回的是一个列表，可以通过列表索引完成）

**5.3卷积权重**

有层了之后我们可以通过layer.get\_weights()获取各层权重，然后再通过plot\_conv\_weights函数来绘制权重图。请学员通过layer.get\_weights()函数获取卷积层1和卷积层2的权重并调用plot\_conv\_weights函数来绘制权重矩阵：

**5.4绘制出卷积层输出的帮助函数**

请参考实验3 相应部分（4.2）完成绘制卷积层输出的帮助函数plot\_conv\_output(values)，其中参数values就代表了卷积层 （比如卷积层1传递给value来完成函数内部操作）

**5.5输入图像**

请参考实验3中的4.3部分来完成plot\_image(image)函数（参数image表示图像）并绘制测试集的第一张图片。

**5.6卷积层输出之方法一**

**5.6.1 基于K函数的模型转换函数**

在Keras模型中有不同的方法来获取图层的输出。方法一是使用 K函数将Keras模型的一部分转换为函数。首先需要导入该函数：

from tensorflow.python.keras import backend as K

请使用K函数来获取图层输出（参数分别是输入层的输入和卷积层1的输出），并赋值给output\_conv1

**5.6.2 获取卷积层1的输出**

然后我们可以用输入图像调用经由K函数转换的output\_conv1函数

请在上一步已经通过得到的output\_conv1函数（该函数的输入是一张图片）的基础上，用图片image1输入到函数中，并取返回的第一个维度的值给layer\_output1。请输出查看一下layer\_output1的形状。

提示该函数的输入是image，它需要被嵌套在2个列表里面即[[image1]]来作为输入(因为改函数需要这个维度的数组)，同时该函数返回一个比我们想要的维度更多的数组，所以我们需要取出第一维的元素返回给layer\_output1。

**5.6.3绘制输出**

最后，我们可以绘制卷积层的所有16个通道的输出。请使用plot\_conv\_output函数绘制卷积层1的输出

**5.7卷积层输出之方法二**

**5.7.1 基于Model函数的图层输出获取**

Keras还有另一种获取模型内层输出的方法。该方法使用与原始模型相同的输入创建另一个功能模型，但输出现在取自我们感兴趣的卷积层（即卷积层2的输出）。

请使用Model函数来获取图层输出（参数分别是输入层的输入和卷积层2的输出），并赋值给output\_conv2

**5.7.2获取卷积层2的输出**

Model函数的输出的是一个新建的模型对象。对于上一步得到的模型对象output\_conv2，我们可以通过该模型对象调用典型的Keras函数。为了得到卷积层的输出，我们通过output\_conv2调用predict()函数，predict()函数的输入是经过numpy数组转换的输入图像image1（predict函数的输入需要np数组）。

请调用output\_conv2的predict()函数得到卷积层2的输出，赋值给layer\_output2并查看其形

**5.7.3绘制输出**

最后，我们可以绘制卷积层的所有36个通道的输出。使用plot\_conv\_output函数绘制卷积层2的输出

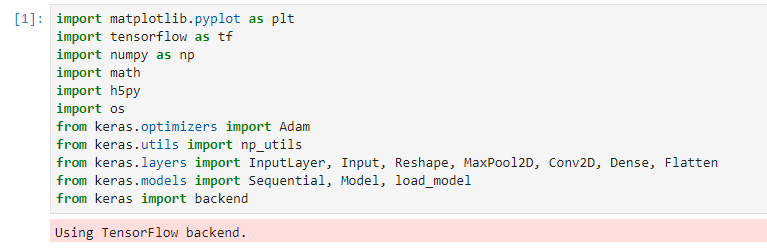
**三、程序清单**

**1．准备工作**

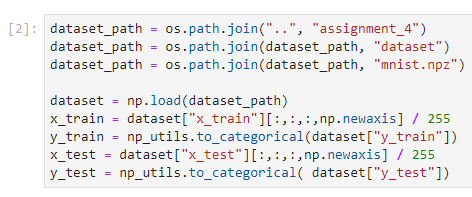
**1.1． 导入包和数据集**

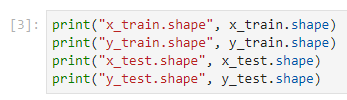
**1.1.1 安装h5py包**

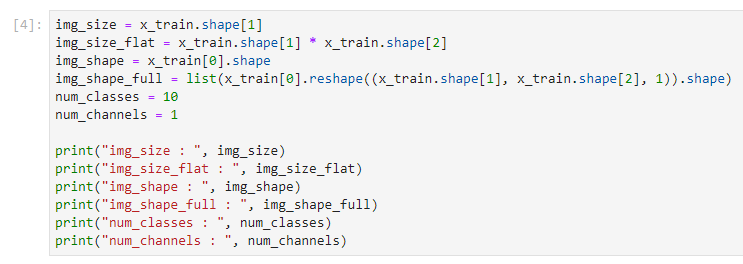
**1.1.2 导入相关包**



**1.2. 载入数据**



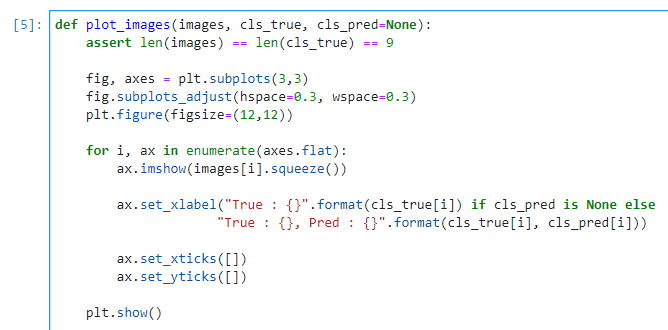




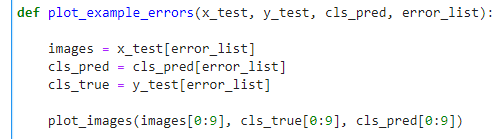
**1.3．配置神经网络**



**1.4.绘制图像的辅助函数**



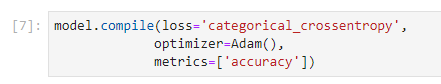
**1.5.绘制错误分类图像的辅助函数**



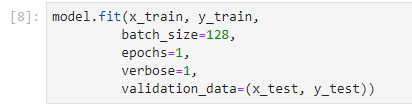
**2.序列模型**

**2.1模型框架**

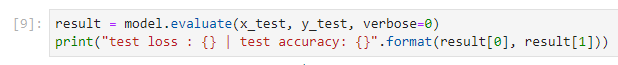
**2.2模型编译**



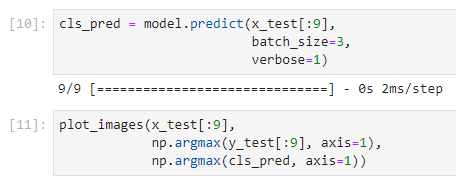
**2.3 训练**



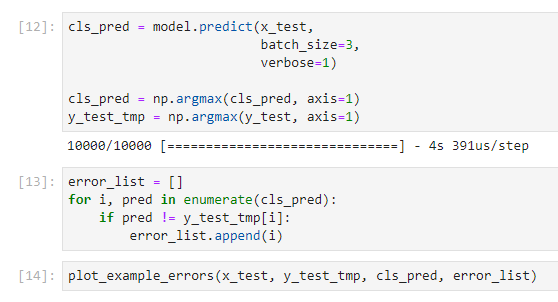
**2.4评估与性能指标**



**2.5 预测**

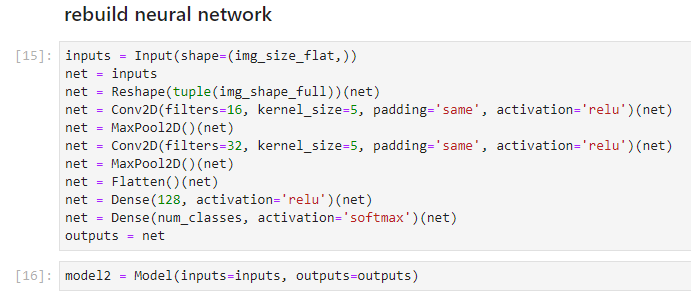


**2.6 错分类的图片**

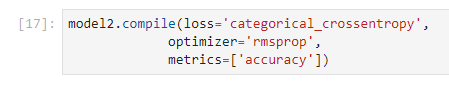


**3. 功能模型**

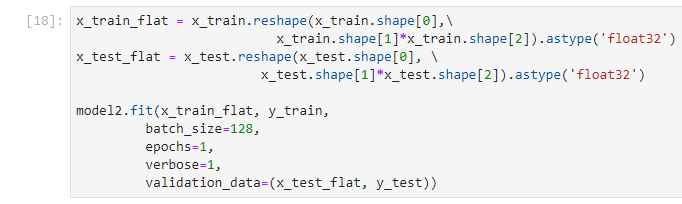
**3.1 模型框架**



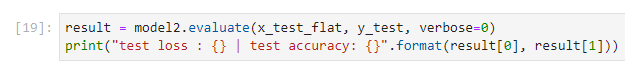
**3.2 模型编译**



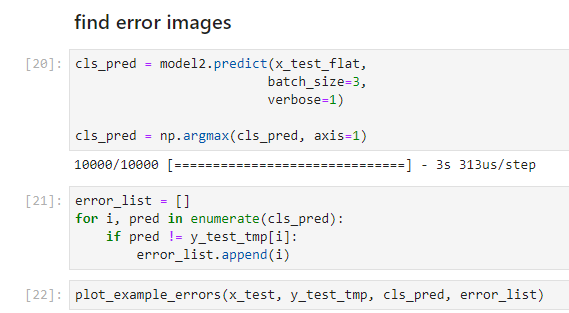
**3.3 训练**



**3.4评估**

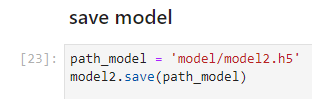


**3.5错分类的图片**



**4. 保存和加载模型**

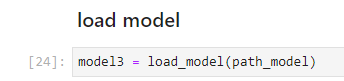
**4.1保存Keras模型**



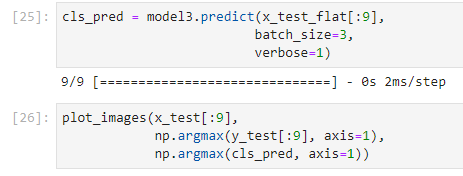
**4.2删除模型**



**4.3加载模型**

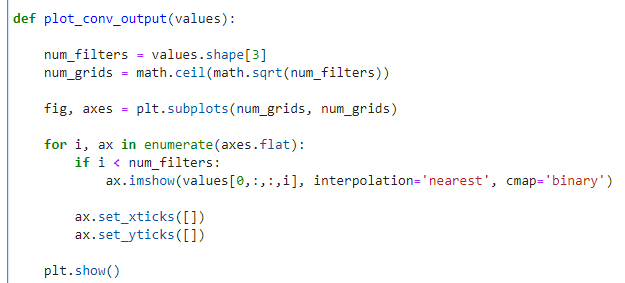


**4.4用加载的模型来预测测试集的前9张图片**

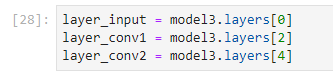


**5权重和输出的可视化**

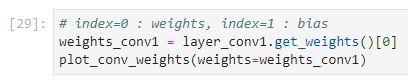
**5.1画卷积权重的辅助函数**



**5.2得到层**

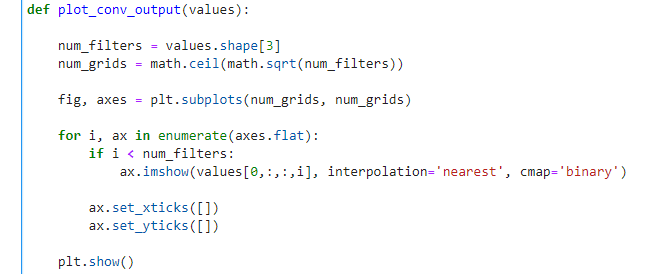


**5.3卷积权重**

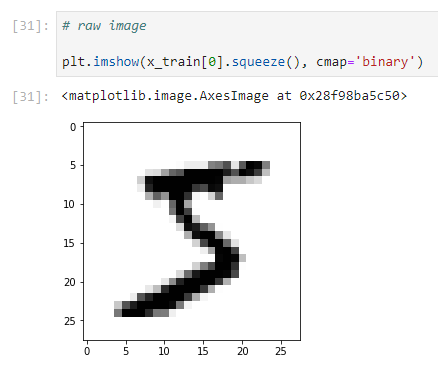




**5.4绘制出卷积层输出的帮助函数**



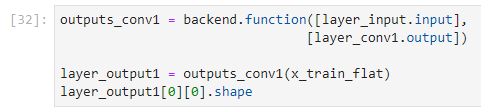
**5.5输入图像**



**5.6卷积层输出之方法一**

**5.6.1 基于K函数的模型转换函数**

**5.6.2 获取卷积层1的输出**



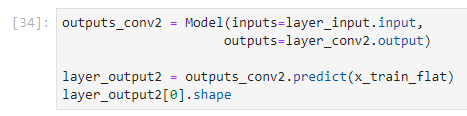
**5.6.3绘制输出**



**5.7卷积层输出之方法二**

**5.7.1 基于Model函数的图层输出获取**

**5.7.2获取卷积层2的输出**



**5.7.3绘制输出**



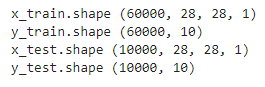
1. **结果分析**

**1．准备工作**

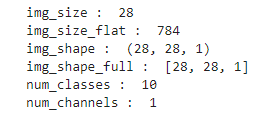
**1.1． 导入包和数据集**

**1.2. 载入数据**

数据集大小：

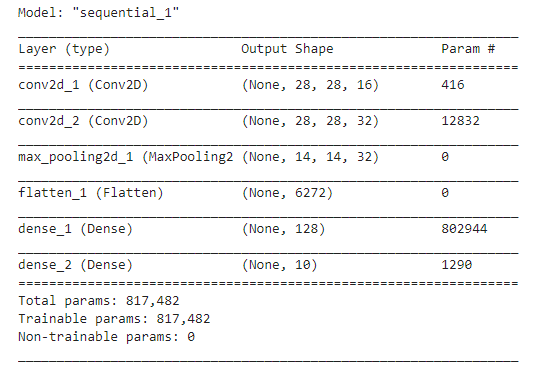


图片的参数：



**1.3．配置神经网络**

神经网络Model结构：



**2.序列模型**

**2.3 训练**

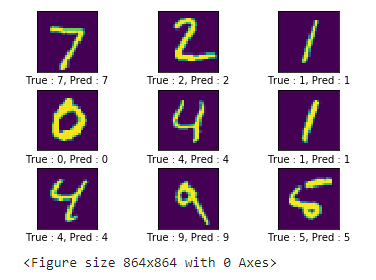
模型训练结果：



**2.4评估与性能指标**

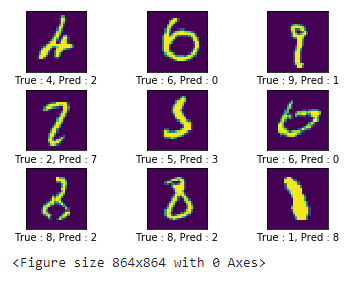
模型测试结果：



 前九张图片的训练结果：

**2.6 错分类的图片**

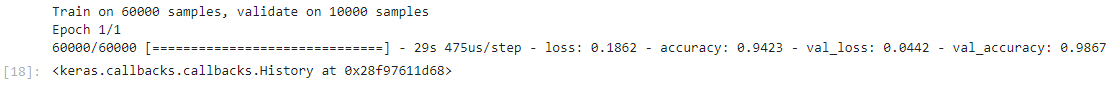
被错误分类的图片：



**3. 功能模型**

**3.3 训练**

模型训练结果：



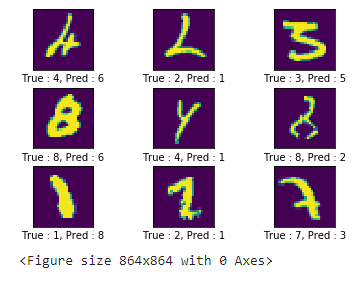
**3.4评估**

模型测试结果：



**3.5错分类的图片**

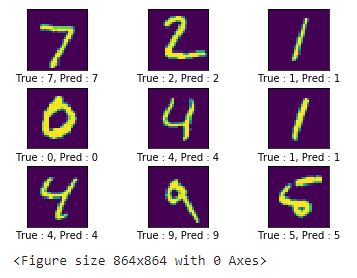
被错误分类的图片：



**4. 保存和加载模型**

**4.4用加载的模型来预测测试集的前9张图片**

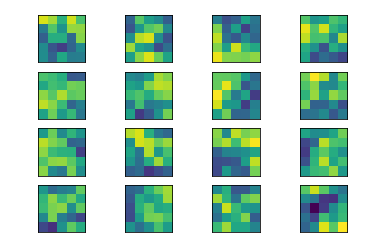
测试结果：



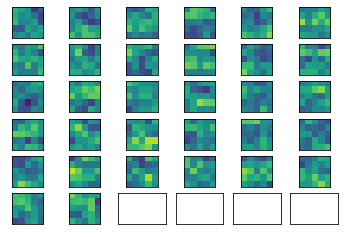
**5权重和输出的可视化**

**5.5输入图像**

第一层卷积层权重：



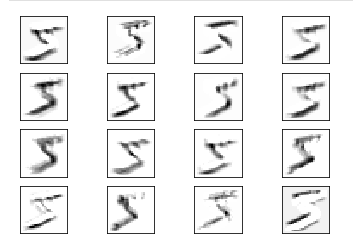
第二层卷积层权重：



**5.6卷积层输出之方法一**

**5.6.3绘制输出**

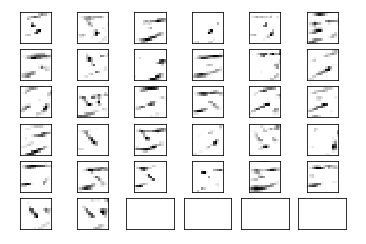
第一层卷积层输出结果：



**5.7卷积层输出之方法二**

**5.7.3绘制输出**

第二层卷积层输出结果：



1. **调试报告**
2. **所遇到的问题**

1. 导包的时候时按照老师的提示通过from tensorflow.python.keras导入的，这样会报错，最后是用from keras导入的。

2. 在绘制卷积网络的单层输入图时，必须先输入的图像整个的数据集，之后再提取第一张图片的输出。

1. **经验和体会**

这次的实验相对于前几次的来说，工作量有所加大，代码行数也在持续上升。同时，自己也在对这方面的知识不断地进行熟悉和巩固。当把卷积神经网络的输出plot出来后，突然理解了神经网络的可视化：其实我们最后一层的数据也可以当作神经网络的可视化。只不过我们只用其来计算损失函数罢了。而这又可以拓展到GAN和实例分割，他们所用到的用来计算Loss的output，也就是我们plot出来的卷积层而不是softmax之后的类别。

源码链接：

<https://github.com/sunnyswag/ml_dl_assignments/blob/master/assignment_5/assignment_5.ipynb>