

物联网中间件技术

实验报告

**姓 名： 胡晗**

**学 号： U201714518**

**班 级： 物联网工程1701**

**指导教师： 顾琳**

**计算机科学与技术学院**

**2020 年 4 月 13 日**

一、实验名称

基于TensorFlow和Keras平台的手写数字识别系统。

二、实验要求

1. 熟悉TensorFlow和Keras平台和提供的库。
2. 利用平台和提供的库，实现以下的功能：
   1. 用Q-learning 的方法实现一个小例子在世界寻找宝藏。
   2. 在MNIST数据集上训练一个简单的深度神经网络，改变神经元的个数和迭代的次数，考察训练的神经网络的准确度的变化。
   3. 保存应用训练好的的深度神经网络，识别新的手写数字。

三、实验平台搭建

一、TensorFlow：

各种安装方式参见: https://tensorflow.google.cn/install/install\_linux

Installing with Virtualenv

Take the following steps to install TensorFlow with Virtualenv:

1. Install pip and Virtualenv by issuing one of the following commands:

$sudo apt-get install python-pip python-dev python-virtualenv # for Python 2.7

$sudo apt-get install python3-pip python3-dev python-virtualenv # for Python 3.n

（Redhat系列：apt-get换成yum）

1. Create a Virtualenv environment by issuing one of the following commands:

$virtualenv --system-site-packages targetDirectory # for Python 2.7

$virtualenv --system-site-packages -p python3 targetDirectory # for Python 3.n

where targetDirectory specifies the top of the Virtualenv tree. Our instructions assume that targetDirectory is ~/tensorflow, but you may choose any directory.

1. Activate the Virtualenv environment by issuing one of the following commands:

$ source ~/tensorflow/bin/activate # bash, sh, ksh, or zsh

$ source ~/tensorflow/bin/activate.csh # csh or tcsh

The preceding source command should change your prompt to the following:

(tensorflow)$

1. Ensure pip≥8.1 is installed:

(tensorflow)$ easy\_install -U pip

1. Issue one of the following commands to install TensorFlow in the active Virtualenv environment:

(tensorflow)$ pip install --upgrade tensorflow# for Python 2.7

(tensorflow)$ pip3 install --upgrade tensorflow# for Python 3.n

(tensorflow)$ pip install --upgrade tensorflow-gpu# for Python 2.7 and GPU

(tensorflow)$ pip3 install --upgrade tensorflow-gpu # for Python 3.n and GPU

If the above command succeeds, skip Step 6. If the preceding command fails, perform Step 6.

1. (Optional) If Step 5 failed (typically because you invoked a pip version lower than 8.1), install TensorFlow in the active Virtualenv environment by issuing a command of the following format:

(tensorflow)$ pip install --upgrade tfBinaryURL # Python 2.7

(tensorflow)$ pip3 install --upgrade tfBinaryURL # Python 3.n

where tfBinaryURL identifies the URL of the TensorFlow Python package. The appropriate value of tfBinaryURL depends on the operating system, Python version, and GPU support. Find the appropriate value for tfBinaryURL for your system [here](https://tensorflow.google.cn/install/install_linux%23the_url_of_the_tensorflow_python_package). For example, if you are installing TensorFlow for Linux, Python 3.4, and CPU-only support, issue the following command to install TensorFlow in the active Virtualenv environment:

(tensorflow)$ pip3 install --upgrade \

**二、**Keras:

在安装了TensorFlow的基础上，安装Keras。

1. 源码安装

从源码安装Keras时，首先git clone keras的代码：

git clone https://github.com/fchollet/keras.git

接着 cd 到Keras的文件夹中，并运行下面的安装命令：

sudo python setup.py install

1. 使用PyPI来安装Keras

sudo pip install keras

如果你用的是virtualenv虚拟环境，不要用sudo就好。

安装教程: [http://keras-cn.readthedocs.io/en/latest/#\_2](http://keras-cn.readthedocs.io/en/latest/%2523_2)

四、实验结果

用Q-learning 的方法实现一个小例子在世界寻找宝藏。

1. 实验结果：

（1）参数设置

设置地图大小为4个单元高、4个单元宽，单元大小为40\*40像素。

地图中红色矩形为探险者，黑色矩形为陷阱，黄色圆形为宝藏，白色区域为地面。探险者掉入陷阱奖励-1，找到宝藏奖励+1，其他区域+0

（2）测试截图

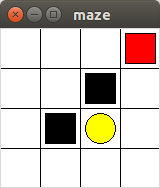


图 1 寻找宝藏测试

（3）修改参数

设置地图大小为10个单元宽，10个单元高。

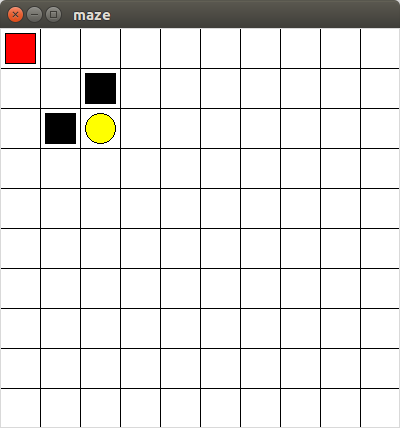


图 2 修改宝藏地图参数

1. 问题和实验感想：

（1）4\*4地图下，测试开始时红色矩形开始对周围区域进行试探，发现为地面时继续向前试探，落入陷阱后下次在相同位置改变行进方向，当发现宝藏后，之后的行进过程参照发现宝藏的路线行进，最开始发现宝藏需要寻找较长时间，每发现一次宝藏，寻找速度逐渐提升，行进路线越来越短。

（2）10\*10地图下，如果红色矩形探测时进入右下方白色区域，需要花比较长的时间进行探测，在寻路过程中，行进方向是随机的，这是由决策方式决定的，实验中对下一次行进方向的选择基于之前的选择的反馈和随机方式。

在MNIST数据集上训练一个简单的深度神经网络。

1. 训练结果：

（1）网络模型图

使用keras工具模块中的plot\_model生成网络模型图：

plot\_model(model, to\_file = 'model.png')

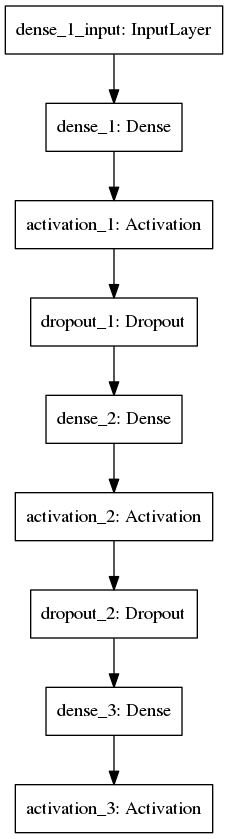


图 3 深度神经网络模型图

（2）网络模型概述

调用序列化模型的summary函数输出定义的模型信息：

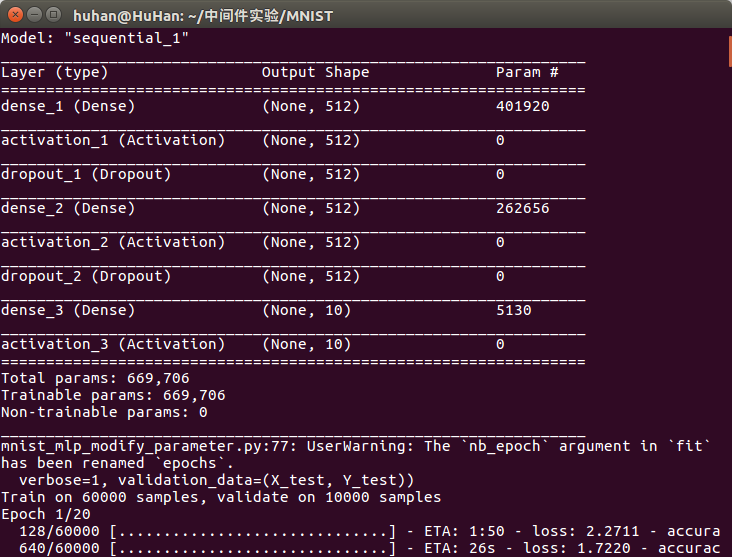


图 4 序列化模型信息

（3）迭代过程

训练、迭代，在控制台输出迭代过程，共迭代20次。

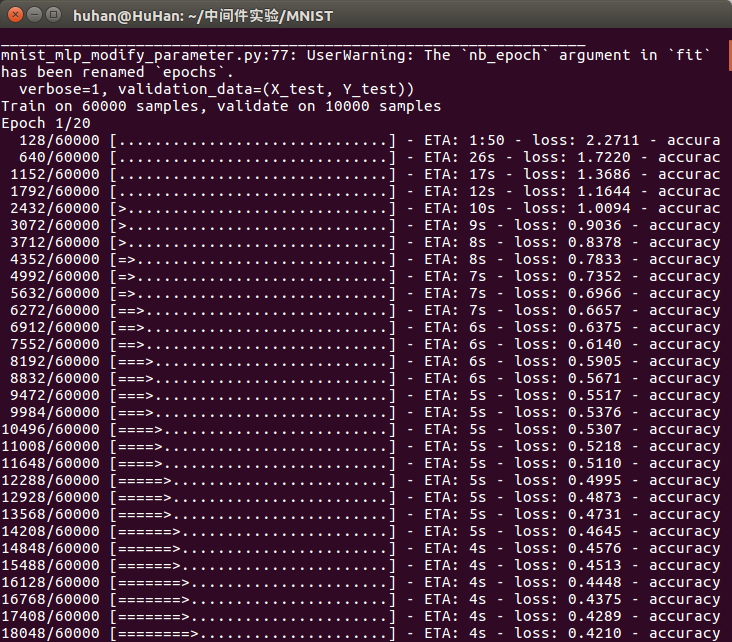


图 5 迭代过程

（4）测试集测试

测试集测试后进行测试结果的评估，评估测试误差和准确率：

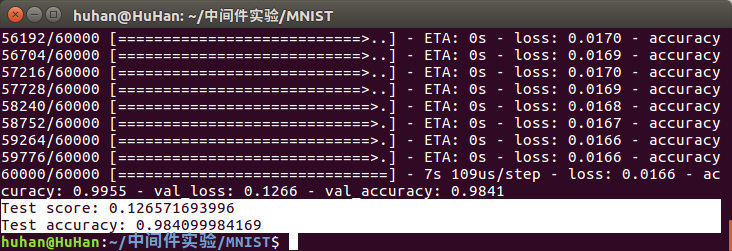


图 6 测试集评估

1. 改变神经元的个数和迭代的次数，考察训练的神经网络的准确度的变化：

（1）原来的模型的准确度和误差变化曲线：

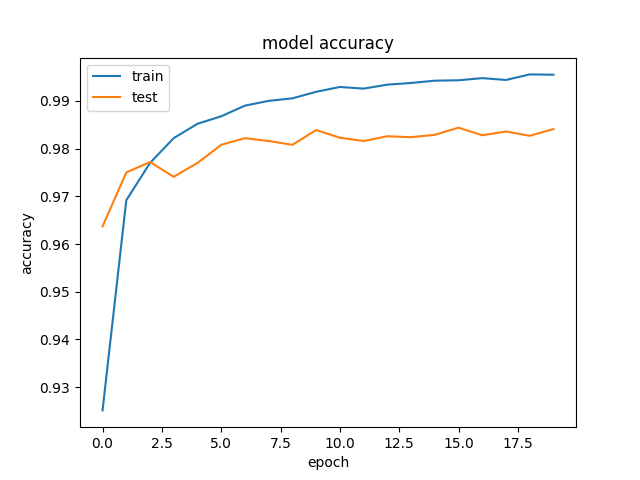


图 7 原模型准确度变化

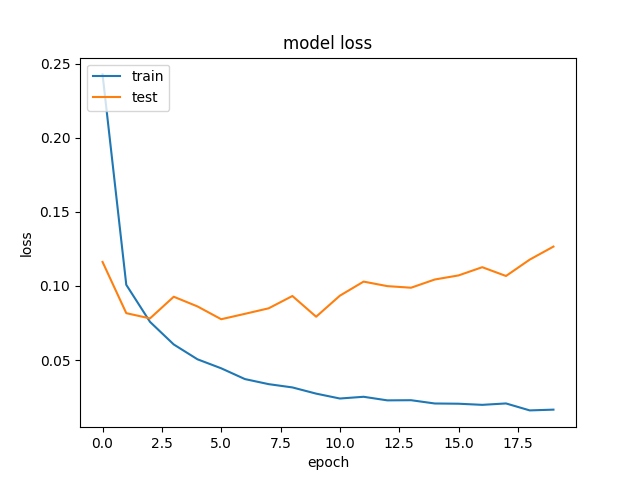


图 8 原模型误差变化

（2）更改迭代次数

更改迭代次数为30次，准确度和误差变化为：

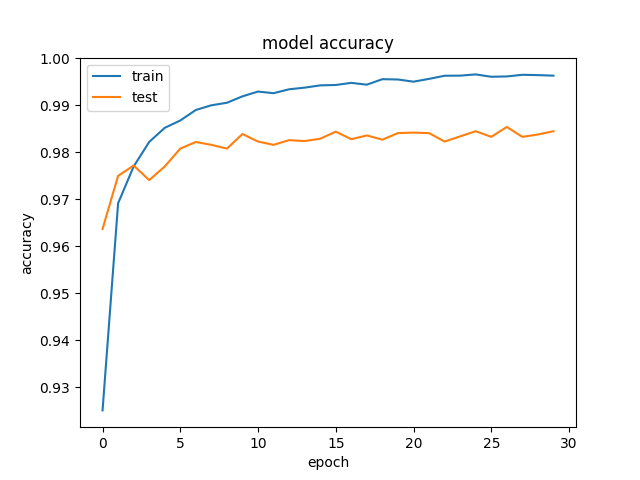


图 9 更改迭代次数后准确率

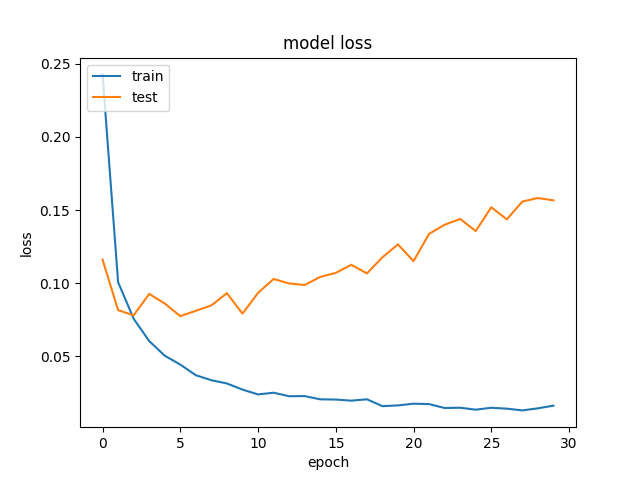


图 10 更改迭代次数后误差

可以观察到，训练准确度和误差无明显变化。测试时准确度和误差都产生更多的波动。而误差在原来迭代20次时最终接近0.13，而迭代30次时超过0.15。

（3）更改神经元个数

更改第一个全连接层为600维输出，第二个全连接层也为600维输出。

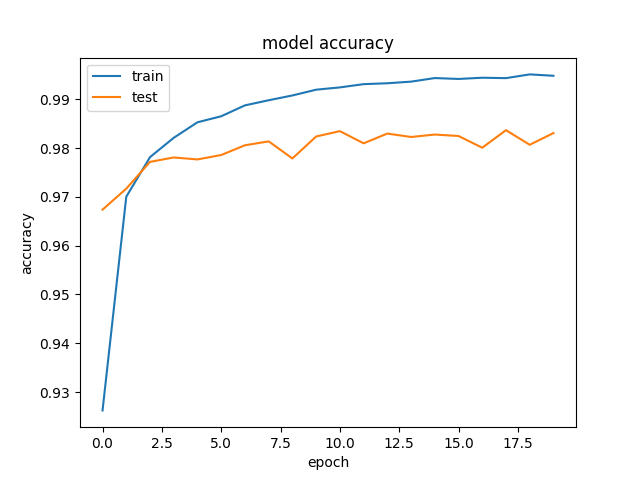


图 11 更改神经元数后准确度

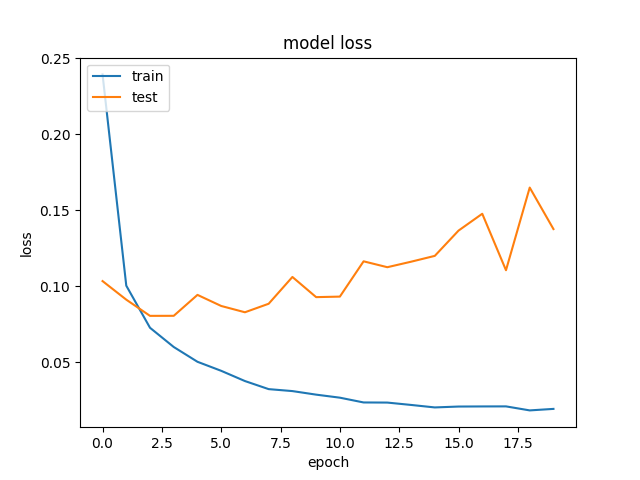


图 12 更改神经元数后误差

观察图像发现测试误差产生大幅度抖动，说明产生了过拟合。

1. 问题和实验感想：

（1）原测试代码中加载数据集是从网络上加载，因为网络原因加载失败，需要将数据集下载到本地后进行加载。

（2）调整深度神经网络参数能够改变神经网络的准确度，调整的方向要保证能提升准确率，同时减少过拟合的影响。

保存应用训练好的的深度神经网络，用于识别新的手写数字。

1. 读取图片的代码和过程：

（1）代码

#读取图片并转换为数组

img = Image.open('./dataset/number6.png')

img\_1 = img.convert('L')

array\_img = np.array(img\_1)

array\_img = array\_img.reshape(1, 784)

print(array\_img)

print('\n')

（2）加载png格式图片，图片内容为数字6：

number6

图 13 数字内容

由于图片为RGB三通道，所以需要转为灰度图像，之后将灰度图像转换为numpy数组，将数组展成784维，转换后的数组为：



图 14 图像转换数组

1. 保存训练好的的深度神经网络的代码和过程：

（1）利用HDF5模块保存和加载模型。

保存模型：

model.save('my\_model.h5')

加载模型：

model = load\_model('my\_model.h5')

（2）保存模型截图

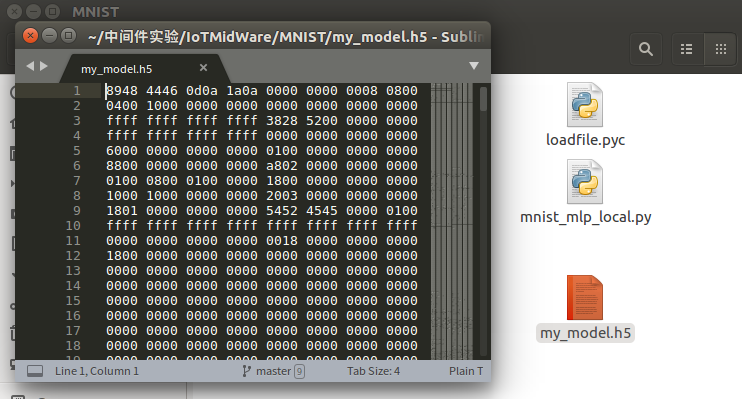


图 15 保存训练模型

1. 识别新的手写数字的结果：

对需要预测的图像数组进行预测：

#预测图片数字

pred = model.predict(array\_img)

#输出预测结果

print('predicted:', pred)

预测后的结果为：



图 16 预测结果

预测结果内容为是否为对应数字位，列表下标为对应数字，结果

[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]

代表数字6，预测结果与输入数字一致。

1. 问题和实验感想：

（1）读取到的图片格式需要进行处理，当图片使用非灰度格式保存时，需要将图片格式转为灰度格式，再使用numpy模块的函数将灰度图片转为数组。

（2）深度神经网络的输入为784个神经元，所以需要将灰度图片转化的数组摊平到784维。

（3）利用该深度神经网络，可以应用于需要进行图像数字识别的领域，比如识别图片中的车牌号，也可以进一步扩展训练的分类集合，如对英文字母的分类识别，则通过训练好的神经网络，可以应用于图片文本识别（OCR）的领域。