

**课 程 实 验 报 告**

**课程名称： 物联网中间件**

**专业班级： 物联网工程1601**

**学 号： U2016148989**

**姓 名： 潘翔**

**指导教师： 顾琳**

**报告日期： 2019.4.8**

**计算机科学与技术学院**

目 录

[1 基于 Keras平台的Q-Learning算法 3](#_Toc1060427425)

[1.1 实验环境 3](#_Toc753555753)

[1.2 实验内容与要求 3](#_Toc1312412509)

[1.3 实验过程与结果 3](#_Toc1482164272)

[2 基于 TensorFlow平台的手写数字识别系统 7](#_Toc1070443955)

[2.1 MNIST Test 7](#_Toc1801227166)

[2.2 核心源码说明 12](#_Toc1560096787)

[2.3 实验体会与总结 14](#_Toc1320979606)

[参考文献 15](#_Toc1164262531)

# 基于 Keras平台的Q-Learning算法

## 实验环境

### 系统环境

OS: Manjaro 18.0.4 Illyria

Kernel: x86\_64 Linux 5.0.7-1-MANJARO

CPU: Intel Core i7-6700HQ @ 8x 3.5GHz

GPU: GeForce GTX 965M

RAM: 7865MiB

### 平台环境

CUDA: V10.1.105

Tensorflow: 1.13.1

## 实验内容与要求

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。

## 实验过程与结果

### Q-learning Test



图1.1 Q-Learning Test Run

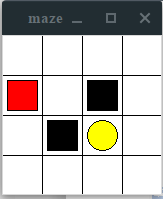


图1.2 Q-Learning Test运行过程

### 更改环境参数



图1.3 Maze属性图

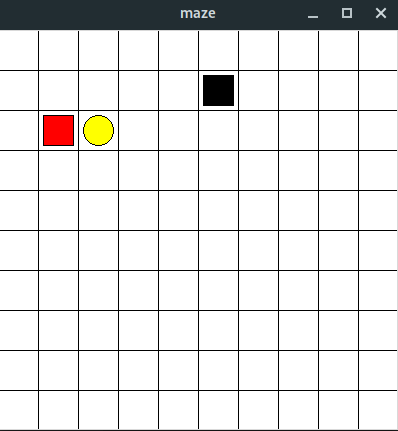


图1.4 Q-Learning Test更改环境参数

表1.1方块状态说明

Red rectangle:          explorer.

Black rectangles:       hells       [reward **=** **-**1].

Yellow bin circle:      paradise    [reward **=** **+**1].

All other states:       ground      [reward **=** 0].

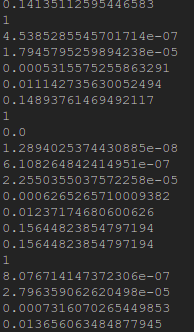


图1.5 Q-Target中间状态输出

可以看到在训练中存在随机性，其中Q-Target在现存Q-Table状态下并非永远最优。

# 基于 TensorFlow平台的手写数字识别系统

## MNIST Test

### 模型训练

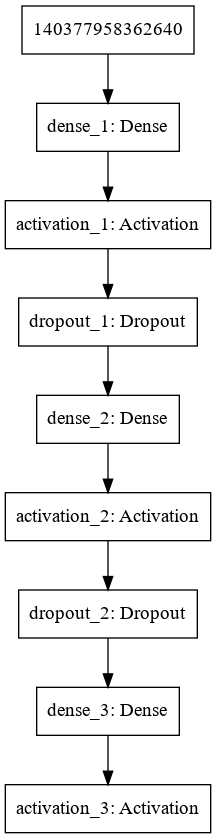


图2.1 CNN网络可视化

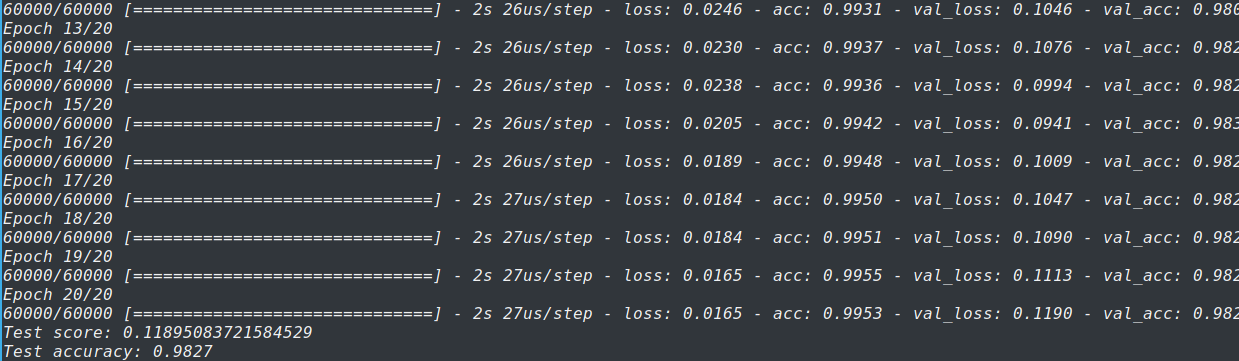


图2.2 MNIST Test运行过程

可以看到在19/20左右Loss不再下降，说明模型已经接近收敛，增大epoch不会有较大提升



图2.3 MNIST Test Tensorboard训练过程可视化

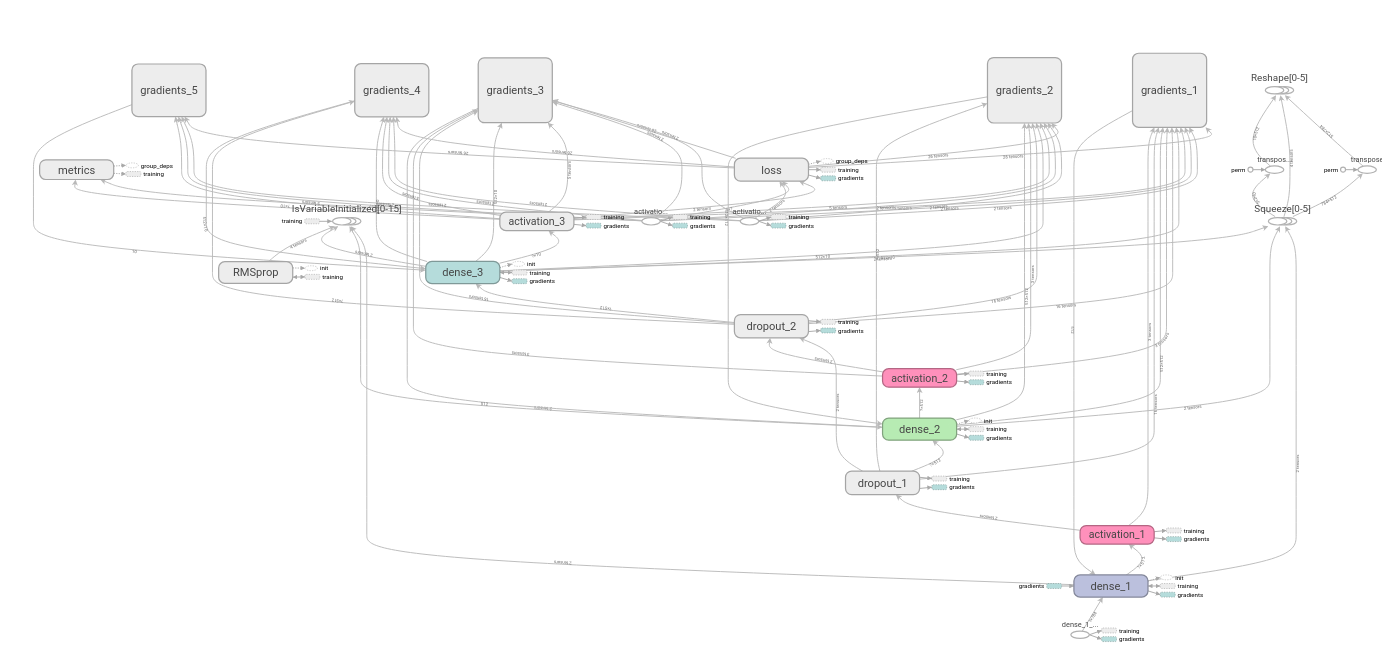


图2.4 MNIST Test Tensorboard计算图可视化

### 保存模型

**from** keras.models **import** load\_model

model.save('mnish.h5')

### 模型测试

对一张MINIST手写数字图片进行求和

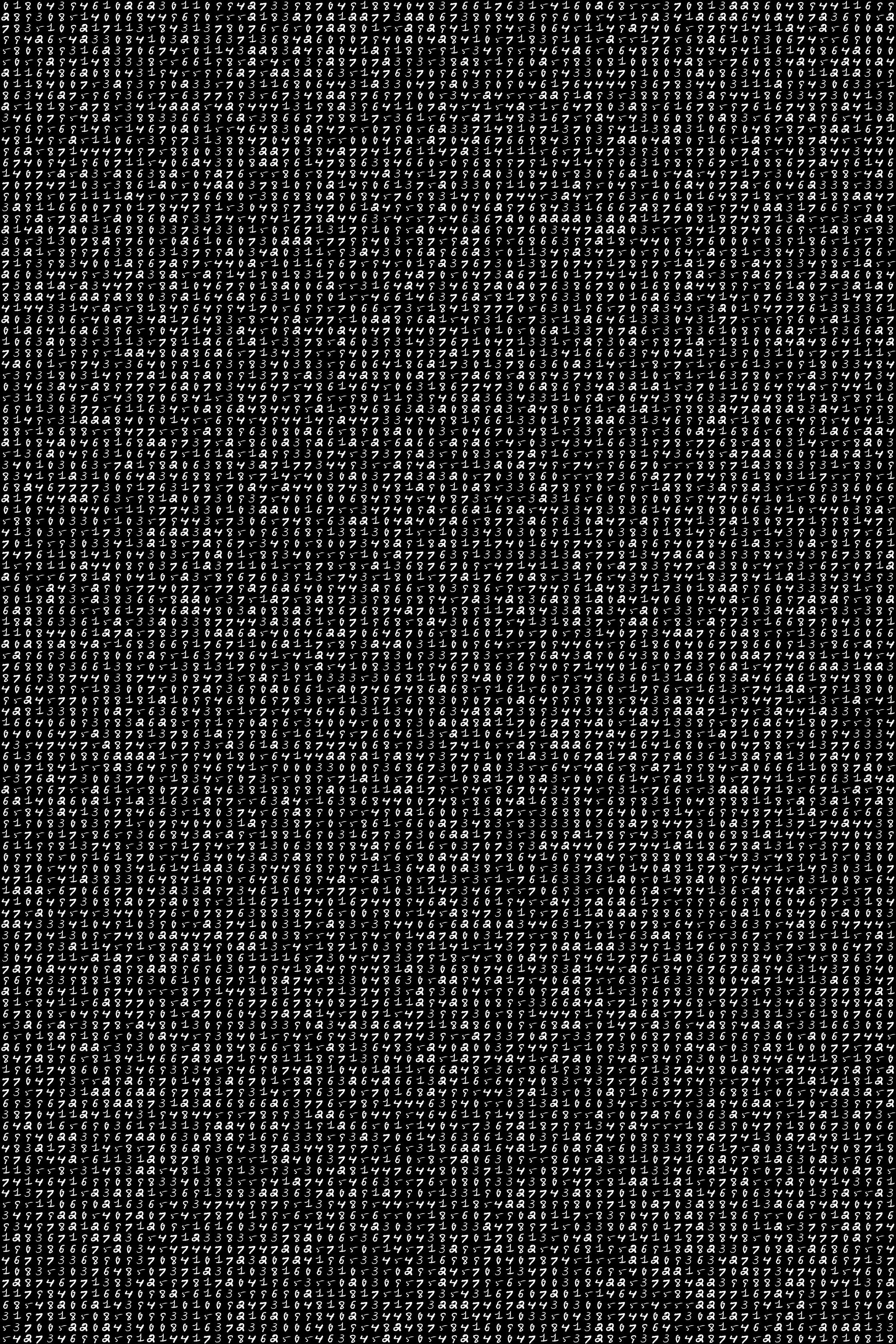


图2.5 MNIST Matrix求和

**import** glob

**import** cv2

src\_img**=**"./number\_matrix.bmp"

count**=**0

img\_src**=** cv2.imread(src\_img)

total**=**0

before\_x**=**0

x**=**0

**for** x **in** range (0,(3360**-**28),28):

    print(x)

**for** y **in** range (0,(2240**-**28),28):

        img\_cut **=** img\_src[x:x **+** 28, y:y **+** 28]

        img **=** cv2.cvtColor(img\_cut,cv2.COLOR\_RGB2GRAY) # RGB图像转为gray

        img **=** cv2.bitwise\_not(img)

        img **=** cv2.resize(img, (28, 28), interpolation**=**cv2.INTER\_CUBIC)

        predict **=** model.predict\_classes(img.reshape(1,28,28,1))

        total **+=** predict

        count **+=** 1

print(total)

经过检测，最终求和结果正确



图2.6 求和测试结果

### 模型更改

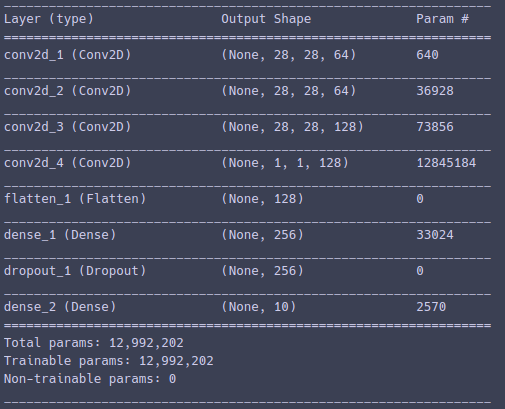


图2.7 Conv2D网络Summary

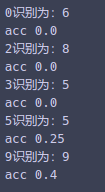


图2.8 Conv2D网络测试结果

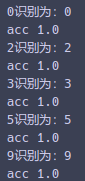


图2.9 二值化后取反测试结果

## 核心源码说明

### Keras-CNN

model **=** Sequential()

callbacks **=** [keras.callbacks.ModelCheckpoint('minist.h5', monitor**=**'val\_acc', verbose**=**1, save\_best\_only**=**True,mode**=**'auto')]

model.add(Conv2D(64, kernel\_size**=**(3, 3),

                 activation**=**'relu',padding**=**'same',

                 input\_shape**=**(28,28,1)))

model.add(Conv2D(64, (3, 3),padding**=**'same', activation**=**'relu'))

model.add(Conv2D(128, (3, 3),padding**=**'same', activation**=**'relu'))

model.add(Conv2D(128, (28, 28),activation**=**'relu'))

model.add(Flatten())

model.add(Dense(256, activation**=**'relu'))

model.add(Dropout(0.5))

model.add(Dense(10, activation**=**'softmax'))

model.summary()

sgd **=** SGD(lr**=**0.01, momentum**=**0.9)

model.compile(loss**=**'categorical\_crossentropy', optimizer**=**sgd, metrics**=**['accuracy'])

### 图片预测

**import** cv2

**import** os

**import** re

file\_path **=** "./test"

path\_list **=** os.listdir(file\_path)

acc**=**0.0

total**=**0

right**=**0

**for** i **in** path\_list:

    image **=** cv2.imread('./test/'**+**i)

    label **=** re.findall('\d+',i)[0]

    img **=** cv2.cvtColor(image,cv2.COLOR\_RGB2GRAY) # RGB图像转为gray

    img **=** cv2.bitwise\_not(img)

    img **=** cv2.resize(img, (28, 28), interpolation**=**cv2.INTER\_CUBIC)

    predict **=** model.predict\_classes(img.reshape(1,28,28,1))

    total**=**total**+**1

    print(str(label)**+**"识别为："**+**str(predict[0]))

**if** int(label)**==**int(predict[0]):

        right**=**right**+**1

    acc**=**float(right)**/**total

    print("acc",acc)

对图像进行二值化处理，然后进行翻转，裁剪最后调用model进行预测

## 实验体会与总结

实验基于Keras实现MINIST,因为对于Keras较为熟悉，实验较为简单，其中有一些细节：

1. 对于输入图片需要注意二值化，同时需要注意，原始训练集为黑底白字，此时可以对于图片中的像素进行统计，判断为白底黑字还是黑底白字，对其适应二值化
2. 图片边缘存在边框可能会有一定影响，需注意裁剪
3. 使用MLP和Conv2D在其上表现类似

最后，感谢实验过程中提供帮助老师和同学以及相应资料的提供者。

# 参考文献

1. 实验指导书
2. Morvan教程:
3. <https://github.com/MorvanZhou>