
华中科技大学

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
1.1 实验环境.....	3
1.2 实验内容与要求.....	3
1.3 实验过程与结果.....	3
2 基于 TENSORFLOW 平台的手写数字识别系统.....	7
2.1 MNIST TEST.....	7
2.2 核心源码说明.....	12
2.3 实验体会与总结.....	14
参考文献.....	15

1 基于 Keras 平台的 Q-Learning 算法

1.1 实验环境

1.1.1 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

1.1.2 平台环境

CUDA: V10.1.105

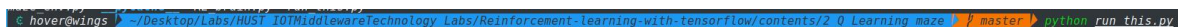
Tensorflow: 1.13.1

1.2 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：
 - a) 用 Q-learning 的方法实现一个小例子在世界寻找宝藏。
 - b) 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。
 - c) 保存应用训练好的深度神经网络，识别新的手写数字。

1.3 实验过程与结果

1.3.1 Q-learning Test



```
e: hover@wings ~$ cd /Desktop/Labs/HUST_107MiddlewareTechnology_Labs/reinforcement-learning-with-tensorflow/contents/2.0_Learning_maze/ ; master$ python run_this.py
```

图 1.1 Q-Learning Test Run

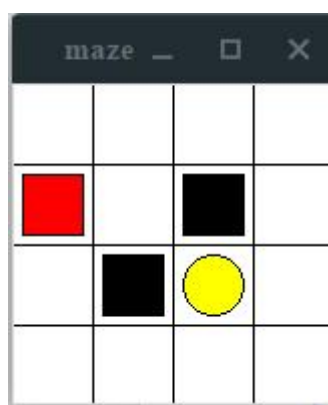


图 1.2 Q-Learning Test 运行过程

1.3.2 更改环境参数

Public Member Functions

```
def __init__(self)
def reset(self)
def step(self, action)
def render(self)
```

Public Attributes

```
action_space
n_actions
canvas
hell1
hell2
oval
rect
```

图 1.3 Maze 属性图

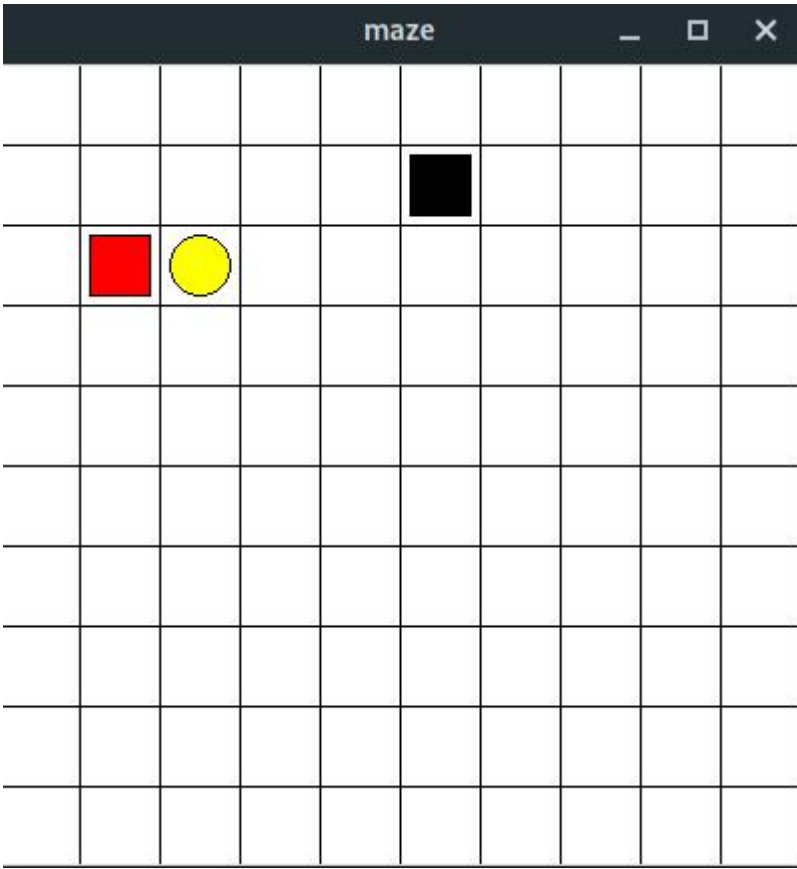


图 1.4 Q-Learning Test 更改环境参数

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].

```
0.14135112595446583
1
4.5385285545701714e-07
1.7945795259894238e-05
0.0005315575255863291
0.011142735630052494
0.14893761469492117
1
0.0
1.2894025374430885e-08
6.108264842414951e-07
2.2550355037572258e-05
0.0006265265710009382
0.01237174680600626
0.15644823854797194
0.15644823854797194
1
8.076714147372306e-07
2.796359062620498e-05
0.0007316070265449853
0.013656063484877945
```

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。

2 基于 TensorFlow 平台的手写数字识别系统

2.1 MNIST Test

2.1.1 模型训练

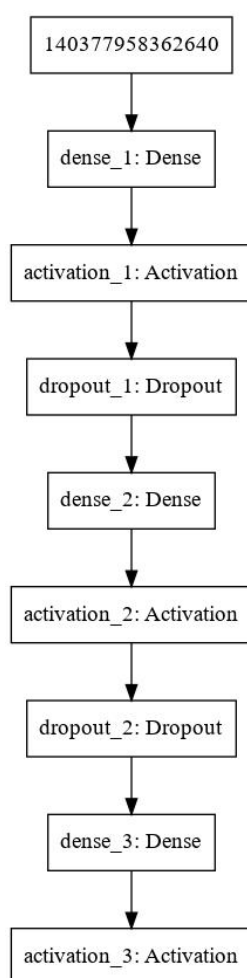


图 2.1 CNN 网络可视化

华中科技大学课程实验报告

```
60000/60000 [=====] - 2s 26us/step - loss: 0.0246 - acc: 0.9931 - val_loss: 0.1046 - val_acc: 0.980
Epoch 13/20
60000/60000 [=====] - 2s 26us/step - loss: 0.0230 - acc: 0.9937 - val_loss: 0.1076 - val_acc: 0.982
Epoch 14/20
60000/60000 [=====] - 2s 26us/step - loss: 0.0238 - acc: 0.9936 - val_loss: 0.0994 - val_acc: 0.982
Epoch 15/20
60000/60000 [=====] - 2s 26us/step - loss: 0.0205 - acc: 0.9942 - val_loss: 0.0941 - val_acc: 0.983
Epoch 16/20
60000/60000 [=====] - 2s 26us/step - loss: 0.0189 - acc: 0.9948 - val_loss: 0.1009 - val_acc: 0.982
Epoch 17/20
60000/60000 [=====] - 2s 27us/step - loss: 0.0184 - acc: 0.9950 - val_loss: 0.1047 - val_acc: 0.982
Epoch 18/20
60000/60000 [=====] - 2s 27us/step - loss: 0.0184 - acc: 0.9951 - val_loss: 0.1090 - val_acc: 0.982
Epoch 19/20
60000/60000 [=====] - 2s 27us/step - loss: 0.0165 - acc: 0.9955 - val_loss: 0.1113 - val_acc: 0.982
Epoch 20/20
60000/60000 [=====] - 2s 27us/step - loss: 0.0165 - acc: 0.9953 - val_loss: 0.1190 - val_acc: 0.982
Test score: 0.11895083721584529
Test accuracy: 0.9827
```

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

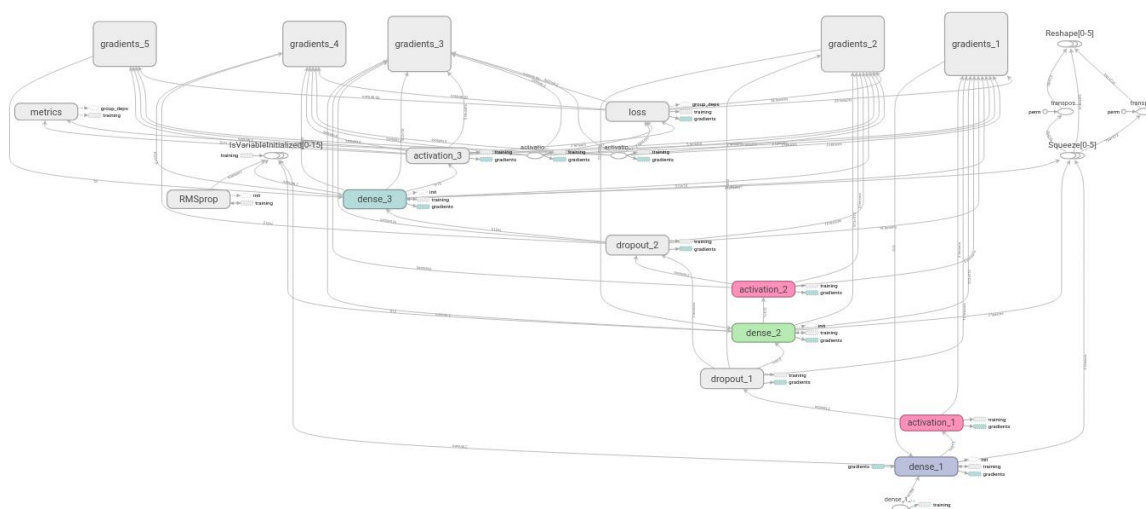


图 2.4 MNIST Test Tensorboard 计算图可视化

2.1.2 保存模型

```
from keras.models import load_model  
model.save('mnish.h5')
```

2.1.3 模型测试

对一张 MNIST 手写数字图片进行求和

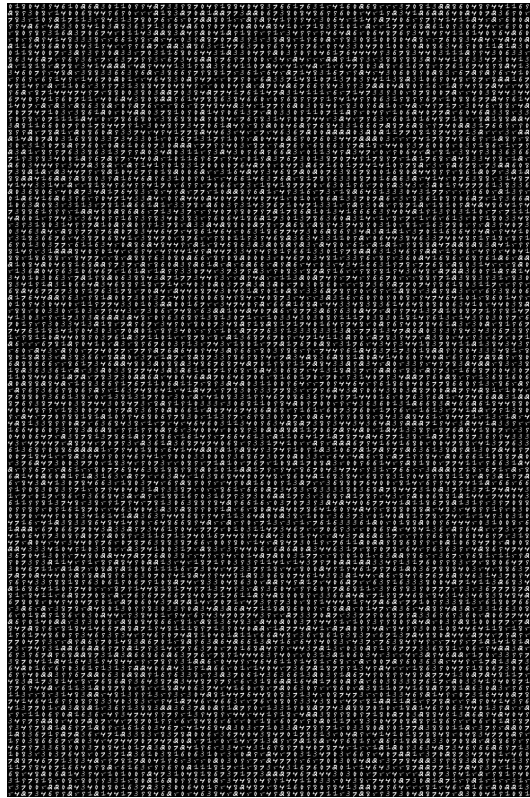


图 2.5 MNIST Matrix 求和

```
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

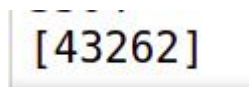


图 2.6 求和测试结果

2.1.4 模型更改

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 64)	640
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
conv2d_4 (Conv2D)	(None, 1, 1, 128)	12845184
flatten_1 (Flatten)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 10)	2570
Total params: 12,992,202		
Trainable params: 12,992,202		
Non-trainable params: 0		

图 2.7 Conv2D 网络 Summary

```
0识别为: 6
acc 0.0
2识别为: 8
acc 0.0
3识别为: 5
acc 0.0
5识别为: 5
acc 0.25
9识别为: 9
acc 0.4
```

图 2.8 Conv2D 网络测试结果

```
0识别为: 0  
acc 1.0  
2识别为: 2  
acc 1.0  
3识别为: 3  
acc 1.0  
5识别为: 5  
acc 1.0  
9识别为: 9  
acc 1.0
```

图 2.9 二值化后取反测试结果

2.2 核心源码说明

2.2.1 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'])
```

2.2.2 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

2.3 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:

<https://github.com/MorvanZhou>