
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 TENSORFLOW 平台的手写数字识别系统.....	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
1.4 核心源码说明.....	7
1.5 思考题.....	ERROR! BOOKMARK NOT DEFINED.
1.6 实验体会与总结.....	8
参考文献.....	9

1 基于 TensorFlow 平台的手写数字识别系统

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

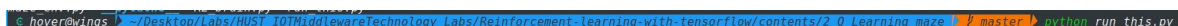
Tensorflow: 1.13.1

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



```
h@hover@wings ~$ cd ~/Desktop/Labs/HUST_IOTMiddlewareTechnology_Labs/Reinforcement-learning-with-tensorflow/contents/2.0_Learning_maze & master$ python run_this.py
```

图 1.1 Q-Learning Test Run

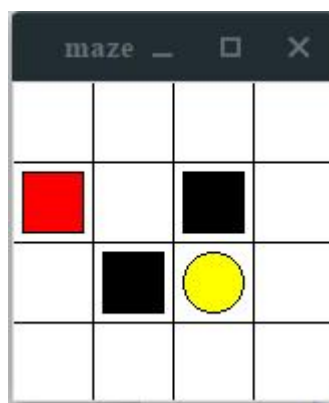


图 1.2 Q-Learning Test 运行过程

1.3.2 MNIST Test

1) 模型训练

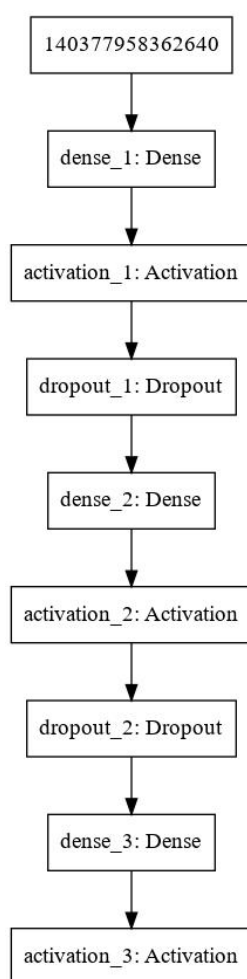


图 1.3 CNN 网络可视化

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 1.4 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

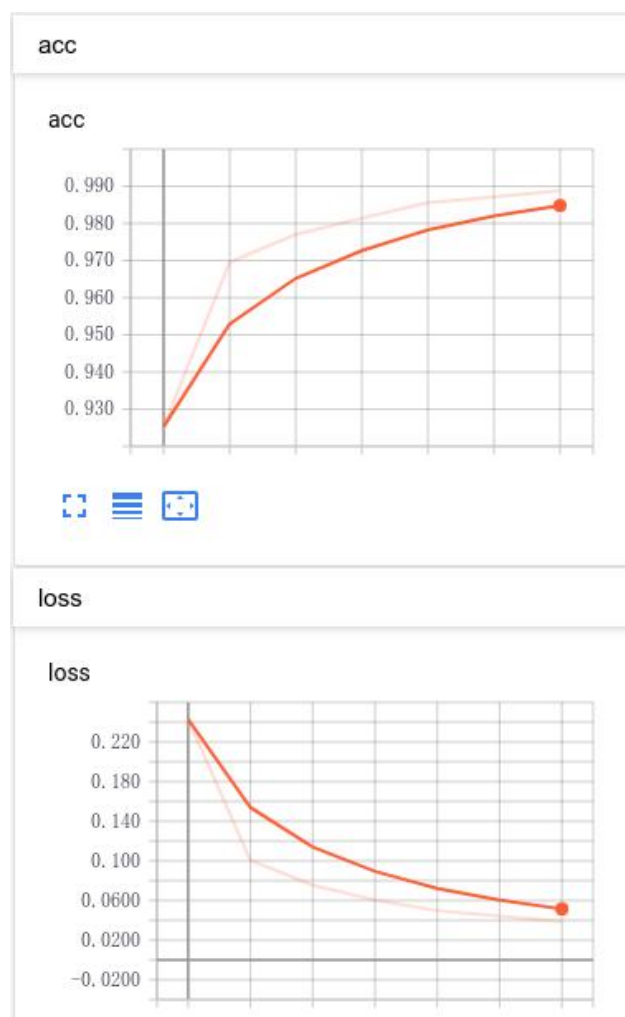


图 1.5 MNIST Test Tensorboard 训练过程可视化

华中科技大学课程实验报告

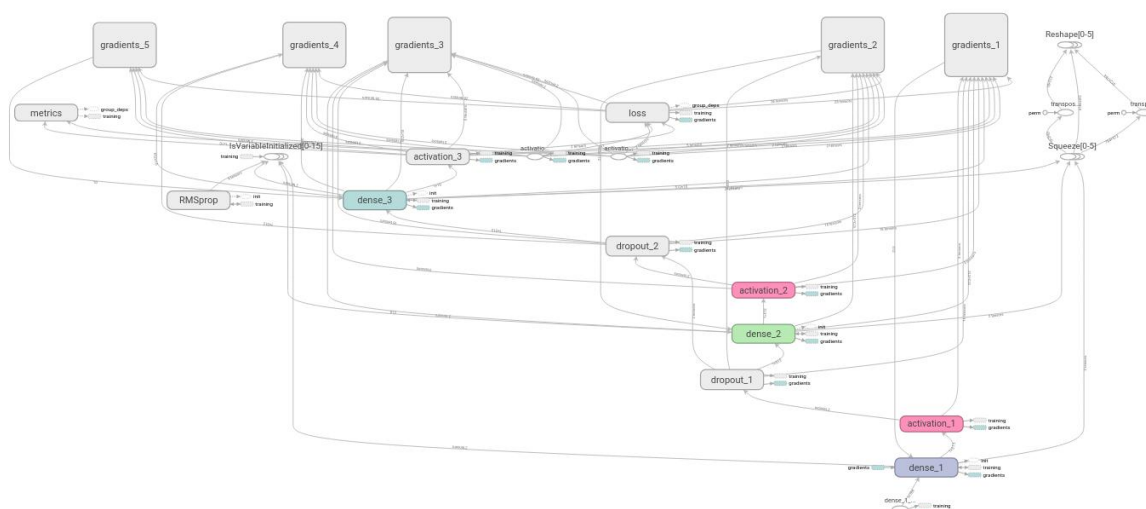


图 1.6 MNIST Test Tensorboard 计算图可视化

2) 保存模型

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

3) 模型测试

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 1.7 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 1.8 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 1.9 二值化后取反测试结果

1.4 核心源码说明

1.4.1 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,
```

华中科技大学课程实验报告

```
momentum=0.9)model.compile(loss='categorical_crossentropy',optimizer=sgd,
metrics=['accuracy'])model.fit(x_train,y_train,
batch_size=128,
epochs=50,
verbose=1,
validation_data=(x_test,y_test),callbacks=callbacks)
```

1.5 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

参考文献

