用深度学习和物联网技术设计“AI魔法棒”

谢作如　浙江省温州中学
邱奕盛　上海人工智能实验室

涉及学科：物理、技术、艺术

在《哈利·波特》系列电影中有各种各样的魔法棒，配合着不同角度的挥动，就可以实现奇特的效果，如点燃火把、拉开窗帘等。从物理学的角度分析，挥动魔法棒其实是改变魔法棒的运动速度，而衡量运动速度变化的物理量是加速度。从理论上讲，只要给魔法棒安装一个加速度传感器，就能感知所有的速度变化。

恰好，掌控板上自带三轴加速度传感器，那么，能否利用掌控板来准确判断魔法棒的手势呢？实际上很多人都尝试过用掌控板的加速度传感器来识别手势，但都因为编程太复杂、难度太大而放弃。目前，加速度传感器仅仅应用在“摇一摇”的判断和x、y轴的角度识别上，有大材小用的感觉。

用加速度识别手势的难度分析

为什么用加速度传感器识别手势很困难呢？因为在魔法棒挥动的过程中，加速度是不断变化的。因此，识别手势需要得到挥动过程中所有的数据变化，需要在手势轨迹中连续采样加速度数值，在得到的一组数据中寻找数据变化的规律。但是，每次人工绘制的手势数据不可能完全相同，总会因为快慢、形状的偏差而产生相近但不相同的数据，这就给编程带来了很大的难度。

在2018年，笔者曾经在本栏目写过《用掌控板体验机器学习》一文，介绍用曼哈顿距离公式来计算两组数据的偏差，即先绘制一次样本数据，作为手势的特征，当新采集的数据与样本数据的手势特征相近，误差小于某个阈值时，就可以识别为正确的手势。然而，这种方式需要人为计算均方根误差，并且不断调整阈值以达到较好的效果，识别算法设计难度高。再加上掌控板的存储和计算资源有限，也很难通过保存多次数据取均值改进识别效果，或者部署AI算法进行识别。

换一种思路来识别手势

1.深度学习

在学习人工智能的时候，大家都知道用机器学习的方式来寻找数据中的关系，是一种非常有效的方式。假设已经拥有一系列不同手势的加速度传感器数据（简称加速度数据集），那么搭建一个BP（Back　Propagation）神经网络（如图1），不断将数据“喂给”这个

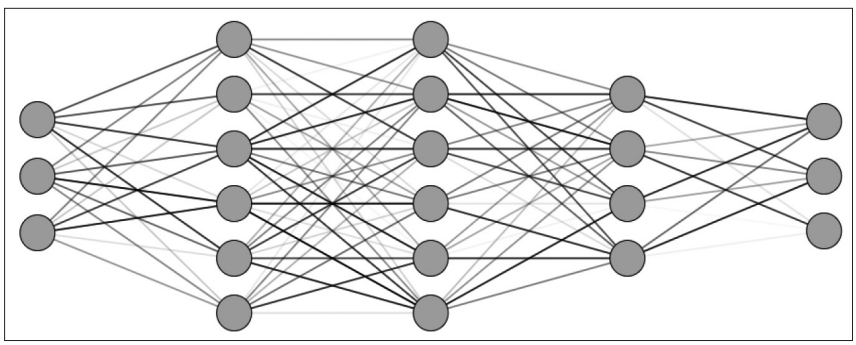

图1

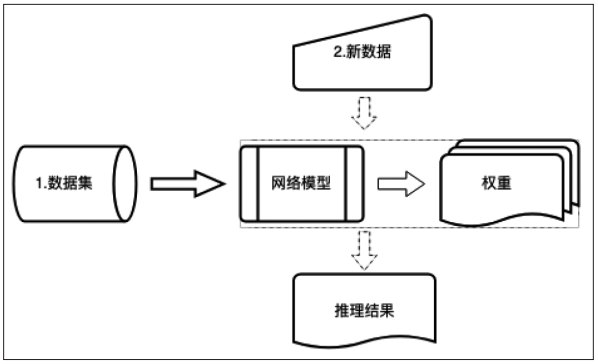

图2

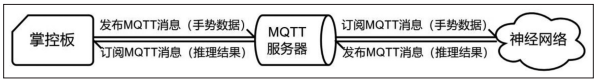

图3

网络模型，就能训练出一个能够识别手势的AI模型。借助于BP神经网络的机器学习，因为拥有神经网络分为多层，属于深度学习中的一种。

深度学习主要分为数据收集、模型训练和模型推理这三个环节。数据收集也就是制作数据集，类似于人类的学习经验知识库，每条数据包含连续的加速度信息x和手势类别信息y。通过收集大量手势数据，并给数据正确标注类别信息，就可以让机器学习这些数据了。模型训练是让深度学习模型通过不断学习数据集信息，达到手势识别的能力。当模型训练好后，只要输入一组数据，就能输出识别的结果了（如图2）。

2.物联网技术

虽然我们没有办法在掌控板上部署一个深度学习的计算框架，但是可以借助物联网技术，将数据传输到PC机上处理。例如，我们可以通过发送MQTT消息将每条数据传输到物联网平台数据库保存，完成手势数据的收集，也可以将需要识别的手势数据，通过MQTT消息传入PC上的深度学习系统中进行推理（手势识别），然后得到结果（如图3）。

● 手势数据的收集

为了手势识别更加准确，笔者规定每一个手势在1秒钟内完成，在这个过程中平均采集128个加速度传感器的数据。采集的程序比较简单，用mPython就能编写。如图4所示，每次按下A键开始采集数据，听到“滴”的一声采集完成，如果确认采集无误，按B键发送到SIoT服务器（消息主题为“shoushi/caiji”）。

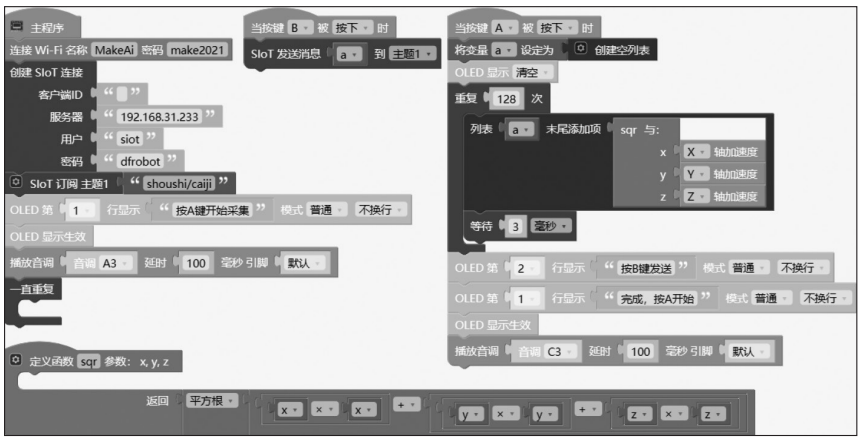

图4

重复多次挥动手中的掌控板，在空中画出相同的路径，如画三角形，这样就可以采集多条数据。当数据达到一定数量（笔者分别收集了100条）时，登录SIoT找到对应主题，可以导出数据到本地。然后更换一个手势，如“×”号，重复以上操作。笔者采集的手势分别是三角形、对号、错号、五角星和圆形。

接下来整理数据，为保存到本地的各种手势数据添加标签，保存为csv文件。文件中每行是一条数据，第一个数据是标签名，第二个数据是标签序号，后面的128个数值是一次完整手势轨迹的加速度值序列。

● 手势模型的训练

1.安装MMEdu

深度学习框架有很多，如Keras、TensorFlow、PyTorch等。MMEdu开发团队基于Pytorch和OpenMMLab，简化了神经网络模型搭建和训练的参数，降低了编程的难度，因此笔者以它为例介绍神经网络模型训练环境的搭建。

通过地址gitee.com/openxlab-edu/OpenMMLab-

```
import numpy as np
filepath='out.csv'
x = np.loadtxt(filepath,dtype=float,delimiter=',',skiprows=0,usecols=range(2,130),encoding='utf-8')
y = np.loadtxt(filepath,dtype=int,delimiter=',',skiprows=0,usecols=(1),encoding='utf-8')
```

图5

```
from MMEdu.MMBase import *
model = MMBase()
model.add_layer('Linear', 128, 64)
model.add_layer('ReLU')
model.add_layer('Linear', 64, 32)
model.add_layer('ReLU')
model.add_layer('Linear', 32, 16)
model.add_layer('ReLU')
model.add_layer('Linear', 16, 5)
model.add_layer('ReLU')
```

图6

```
model.load_data(x, y)
model.train(lr=0.1, epochs=100)
```

图7

```
In [53]:
model.train(lr=0.005, epochs=10)

{epoch:0 Loss:0.6142}
{epoch:1 Loss:0.6141}
{epoch:2 Loss:0.6141}
{epoch:3 Loss:0.6141}
{epoch:4 Loss:0.6141}
{epoch:5 Loss:0.6141}
{epoch:6 Loss:0.6141}
{epoch:7 Loss:0.6140}
{epoch:8 Loss:0.6140}
{epoch:9 Loss:0.6140}
```

图8

Edu可以下载项目文件。解压后即可使用，在Demo文件夹中能够找到范例代码。MMEdu内置了Pyzo，但笔者推荐使用jupyter作为IDE。

2.训练数据模型

根据数据格式，笔者将第二列作为类别信息y，后面的128个数组成的序列作为加速度数据x，以图5所示的方式读入。

接下来，搭建一个BP神经网络模型进行机器学习，神经网络模型的输入层为128个神经元，因为每条数据有128个特征，输出层为5，所以手势共有5类。中间有3个隐藏层，神经元个数分别为64、32、16，隐藏层的作用是增加模型特征提取的效果。隐藏层层数和神经元个数可以自行调整。每个隐藏层后面紧跟一个激活层，用于增强模型的非线性拟合能力，代码如图6所示。

然后将数据载入到网络中，进行训练。这里的lr指的是学习率，epochs是训练轮数，可以根据实际情况修改（如图7）。每训练一轮，会输出一次训练情况，其中Loss越小，表明学习的效果越好。

如果想在此基础上再继续训练，可以重复执行model.train，也可以修改其中的参数lr和epochs继续训练，以达到较好的效果（如图8）。

当效果满意时，也就是当Loss数字不再明显下降时，就可以停止训练，执行model.save("mynet.pkl")来保存模型。

AI魔法棒的部署和测试

笔者将训练好的模型部署在计算机上，将从SIoT的主题shoushi/caiji收到的信息传入神经网络进行推理，根据推理结果决定是否发送消息给SIoT的主题shoushi/jieguo。代码编写如图9所示。

在掌控板原来代码的基础上，添加一段代码，使得在掌控板上也能查看识别的结果（如第108页图10）。

```
import numpy as np
from MMEdu.MMBase import *
import siot
import json

SERVER = "192.168.31.233"
CLIENT_ID = ""
IOT_pubTopic1 = "shoushi/caiji"
IOT_pubTopic2= "shoushi/jieguo"
IOT_UserName = "siot"
IOT_PassWord = "dfrobot"

class_list=["三角形","对号","错号","五角星","圆形"]
model = MMBase()
model.load("mynet.pkl")
def sub_cb(client, userdata, msg):
    decodejson = [json.loads(msg.payload)]
    x_input=torch.tensor(decodejson,dtype=torch.float32)
    pred=model.inference(x_input)
    pred=pred.detach().numpy()
    ans = np.argmax(pred, 1)[0]
    siot.publish(IOT_pubTopic2, class_list[ans])

siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord)
siot.connect()
siot.subscribe(IOT_pubTopic1, sub_cb)
siot.loop()
```

图9

（下转第108页）