人工智能课程报告

任可欣 2015202037

【摘要】本次报告将总结大作业“智能小车”的制作，实现智能小车的自动避障功能，并针对智能小车的一些研究与开发：如何实现智能小车的黑线循迹、蓝牙端口与电脑的连接与基于 MNIST 数据库的手写数字字符识别原理的学习。

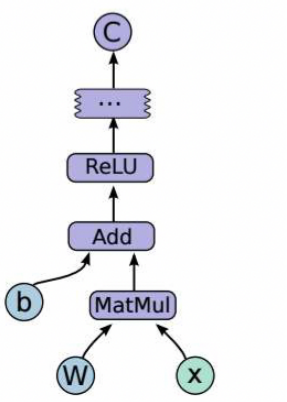
实验背景

1. Arduino

Arduino 是一款便捷灵活、方便上手的开源电子原型平台。包含硬件（各种型号的 Arduino 板）和软件（Arduino IDE）。具有类似 Java、C 语言的开发环境，可以通过运行代码实现对电路板的控制，从而使得小车达到相应的工作效果。

2. Tensorflow 深度学习平台

TensorFlow采用数据流计算，可以由一个有向图表示其数据流计算。在一幅Tensorflow图中，每个节点有一个或者多个输入和无或者多个输出，表示一种操作的实例化。叶子结点通常为常量或者变量，非叶子结点为一种操作，箭头代表的是张量的流动方向。（常量、变量以及节点计算出的结果均可被视为张量）



实验内容

1. 智能小车实验

本次智能小车实验部分大致分为三个阶段：小车搭建、小车自动避障与蓝牙控制、小车摄像与字符识别。

a) 小车搭建

在本次实验中我们使用到了以下器材：

对于 Arduino 板，我们使用到的接口有：接收来自计算机指令信息并与蓝牙串口模块上的 TXD 连接的 digital pin 0(RX)；发送指令信息给计算机，与蓝牙串口模块上的 RXD 相连的 digital pin 1(TX)；以及电源的正负极接口 VCC 和 GND 。通过以上的四个接口控制蓝牙并为 Arduino 板和电机供电。

在了解了 Arduino 板的端口工作原理之后，我们对小车进行了组装。因为长时间缺少对于硬件的接触，在组装过程中我们遭遇了许多问题：小车接触不良，电机驱动板不灵敏，小车马达驱动失败，小车不稳定等。

b) 小车自动避障与蓝牙控制调试

小车自动避障靠的是小车通过传感器，识别前方是否有物体进而根据方位改变小车左轮右轮的运行速度来改变小车的前进方向。在本阶段我们使用的代码基础是老师在 GitHub 中分享的代码，根据自己小车的情况对代码做了相关修改。

一开始我们发现代码编译始终通不过，但是导入的代码语句本身是没有问题的，在一番搜索之后我们发现，在编译的时候缺少头文件“AFMotor.h”，在自行下载了相关头文件代码之后便修复了这个问题。

将代码烧制到 Arduino 板之后，经过马达的驱动，我们的小车终于跑了起来，但是因为小车本身制造问题我们的小车跑起来非常的不稳定，并且只向一个方向绕圈。在经过多次调试以后，使得两轮的转速相当，能够按照一定的方向直线行走。

在自动避障的部分，我们一开始经历了小车不能识别的问题，后来经过电路重新的检查发现我们在面包板上并没有将小车很好的连入电路造成的。在更改传感器的电路连接之后，小车终于可以识别物体，并返回相应距离。在距离小的地方能够停下来更改方向之后，终于达到了我们所要达到的小车自动避障功能。并且通过手机上的 app，能够通过指令给小车传递讯息控制小车的前后左右行进。

c) 扩展功能：黑线循迹

在我们购买的工具包中，有相关的光明电阻与传感器，根据相关手册，在此次人工智能小车的组装中，一并实现了黑线循迹功能：小车会跟随着黑色的轨道前进，达到一定意义上的轨道自主运行。

黑线循迹的原理为：在车的前端安装两个光感传感器，当识别到是黑色的时候表现为熄灯，在白色地砖上表现为亮灯。在这个实验中，一开始我们使用手指作为探测，导致对于传感器的识别产生了误解；之后因为购买黑色胶带无果，使用红色胶带作为替代但是灵敏度不够；最后终于换成黑色胶带，在经过对轨道宽度的调整之后最终确定了实际宽度。

在本次实验中，我们遇见的许多问题均为硬件不灵敏或者对器材了解不够导致的，最终经过不断调试后，终于构成了能够使得小车稳定运行的轨道。

d) 扩展功能：小车摄像与端口连接

i. 在摄像上我们一开始使用了老师给的 Opencv 进行调试，但是发现传送并不稳定并且只有一台手机可以用来传送图像。因此更改了软件为 Microsoft 的 SilverLight 用于在小车上搭载传输摄像的工具。

ii. 在本部分的计划中，图像识别处理我们使用电脑完成，完成以后会将数字传输给小车并对小车进行控制。因此为了将小车蓝牙与电脑连接起来，我们

利用了 PyBluez 提供的 BluetoothSocket 接口进行电脑与小车蓝牙之间的连接，从而实现电脑对小车的遥感控制。

iii. 安装Pyserial库：Pyserial是协助Python控制串口通讯的第三方库。（pyserialV2.7 下载地址为:http://sourceforge. net/projects/pyserial/，pyserialV3.0下载地址为:https:// github.com/pyserial/pyserial。 ）便可以调用相关库函数启用串口监视器，对相关COM口传送特定的信息。实现对于Arduino板相关串口的python编程。

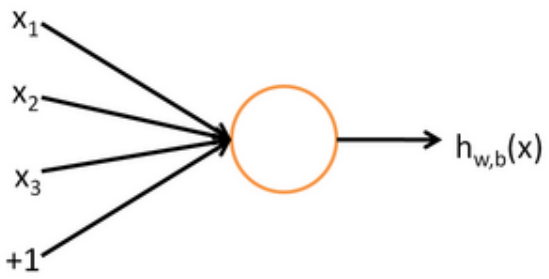
e) 字符图像识别

我们将会使用小车搭载的摄像工具对于前方的字符进行图像识别并由识别到的数字传送回小车蓝牙进而控制小车。

根据资料查找，手写字符的识别方法有多种，如SVM支持向量机、神经网络、KNN、朴素贝叶斯方法等。其中神经网络作为一种经典的模式识别工具，应用广泛。将神经网络应用于手写字符识别，具有识别速度快、分类能力强、有较好的容错性能和学习能力的优点。并且 TensorFlow 提供了构建神经网络的接口，因此便于构建神经网络，简化编程任 务。与传统平台构建的识别模型相比，提高了效率。因此在此对于神经网络实现字符识别做一个总结。

(一)卷积神经网络的原理

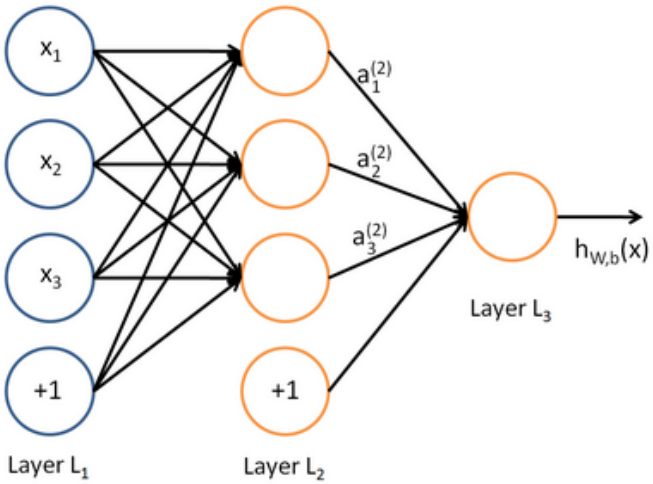
1. 神经网络的每个单元可以如图所示：



对应以下公示：

$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$$

其中，该单元也可以被称作是Logistic回归模型。当将多个单元组合起来并具有分层结构时，就形成了神经网络模型。下图展示了一个具有一个隐含层的神经网络。



其对应如下公式：

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

比较类似的，可以拓展到有2,3,4,5，…个隐含层。

神经网络的训练方法也同Logistic类似，不过由于其多层性，还需要利用链式求导法则对隐含层的节点进行求导，即梯度下降+链式求导法则，专业名称为反向传播。

2. 卷积神经网络

卷积神经网络与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层构成的特征抽取器。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。

在CNN的一个卷积层中，通常包含若干个特征平面(featureMap)，每个特征平面由一些矩形排列的的神经元组成，同一特征平面的神经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值（卷积核）带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。子采样也叫做池

化（pooling），通常有均值子采样（mean pooling）和最大值子采样（max pooling）两种形式。子采样可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。卷积神经网络的基本结构如图所示：卷积神经网络由三部分构成。第一部分是输入层。第二部分由 n 个卷积层和池化层的组合组成。第三部分由一个全连结的多层感知机分类器构成。

在本实验中使用的是BP神经网络：

(二)BP 神经网络设计

BP 神经网络是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。学习规则使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。由输入层、隐含层和输出层组成，层与层之间采用全连接方式，同层之间不存在相互连接。

1. BP 神经网络

三层 BP 神经网络模型所示，分为输入层、隐含层和 输出层，还包括的参数有:

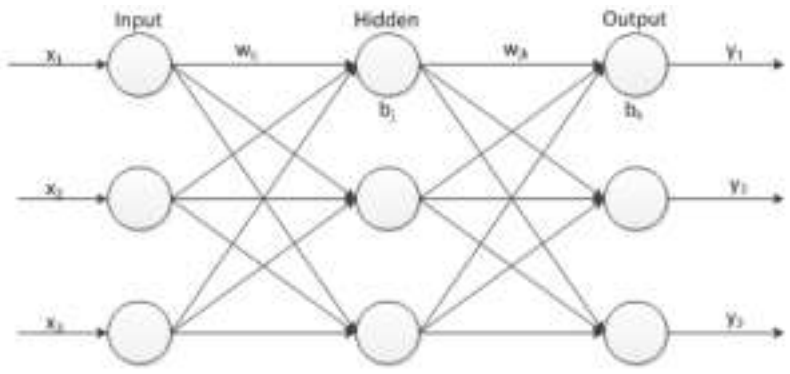
w_{ij} :输入层第 i 单元到第一隐含层第 j 单元的权重;

w_{jk} :隐含层第 j 单元到输出层第 k 单元的权重;

b_j :隐含层第 j 单元激活阈值;

b_k :输出层第 k 单元激活阈值;

f(X):激活函数采用S型激活函数。



误差计算:方差代价函数为 E(w,b) ，其中 y_i 为计算输出结

果，di 为期望输出结果。

$$E(w,b)=\frac{1}{2}\sum_{i=0}^{n-1}(y_i-d_i)^2$$

权值阈值修正:权值和阈值的修正依赖于误差信号的反向传输。根据梯度下降算法，通过计算输出层误差，可以调整隐含层和输出层之间的权值和阈值，同样可以调整输入层和隐含层之间的权值和阈值。通过反复修正权值和阈值，使代价函数 E(w, b) 达到最小。

设节点i和节点j之间的权值为 w_{ij} ，节点j的阀值为 b_j ，每 个节点的输出值为 x_i ，则调整过程为:

$$w_{ij}=w_{ij}-\eta_1\frac{\partial_{E(\omega,b)}}{\partial_{\omega_{ij}}}=w_{ij}-\eta_1\sigma_{ij}x_i$$

$$b_j=b_j-\eta_2\frac{\partial_{E(\omega,b)}}{\partial_{b_j}}=b_j-\eta_2\sigma_{ij}$$

2. 基于TensorFlow实现

本实验采用 **MNIST 手写字符数据集**，手写字符为 28×28 像素的手写数字灰度图像。文件夹中的60000幅手写字符数据，55000幅作为训练集，5000幅作为验证集。测试集有10000幅图像的字符和标签。

输入层设计:手写字符每一张图片的大小为32×32，一维化后，每一张图片作为输入时需要784个输入层神经元节点，其中None表示输入图片的数目:

```
x = tf.placeholder("float", shape=[None, 784])
```

隐含层设计:previous_units为前一层神经元节点数，hid- den_units为当前层神经元节点数:

```
weights = tf.Variable(tf.truncated_normal
```

([previous_units, hidden_units], stddev=1.0 / math.sqrt(float (IMAGE_PIXELS))),name='weights')

biases = tf.Variable(tf.zeros([hidden_ units]),name='biases') hidden = tf.nn.relu(tf.matmul(images, weights) + biases)

输出层设计:输出层为线性的softmax回归模型，用大小为10的一维张量表示 10 个不同的类别:

weights = tf.Variable(tf.truncated_normal ([hidden2_units, 10],stddev=1.0 / math.sqrt(fl oat(hidden2_units))),name='weights')

biases = tf.Variable(tf.zeros([NUM_CLS SES]), name='biases')

logits = tf.matmul(hidden2, weights) + biases

计算损失:以交叉熵作为损失函数，训练过程与方差代价函数类似，但是可以克服方差代价函数更新权重过慢的问题:

cross_entropy = tf.nn.softmax_cross_entr

opy_with_logits(logits, onehot_labels,name= 'xentropy')

loss = tf.reduce_mean(cross_entropy, name='xentropy_mean')

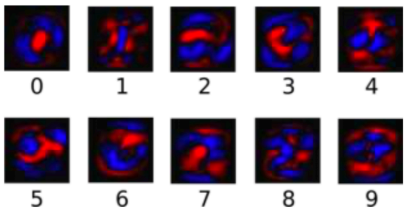
(三)模型训练与评估

TensorFlow 可以灵活访问整个计算图，它可以使用自动微分法找到对于各个变量损失的梯度值。TensorFlow 有大量内置的优化算法 ，采用最速下降法让交叉熵下降，步长为 0.01:

optimizer = tf.train.GradientDescentOpti

mizer (learning_rate)

optimizer 在运行时会使用梯度下降来更新参数。因此，整个模型的训练可以通过反复地运行 optimizer 来完成。 下图显示了模型学习到的图片上每个像素对于特定数字类的权值。红色代表负数权值，蓝色代表正数权值。

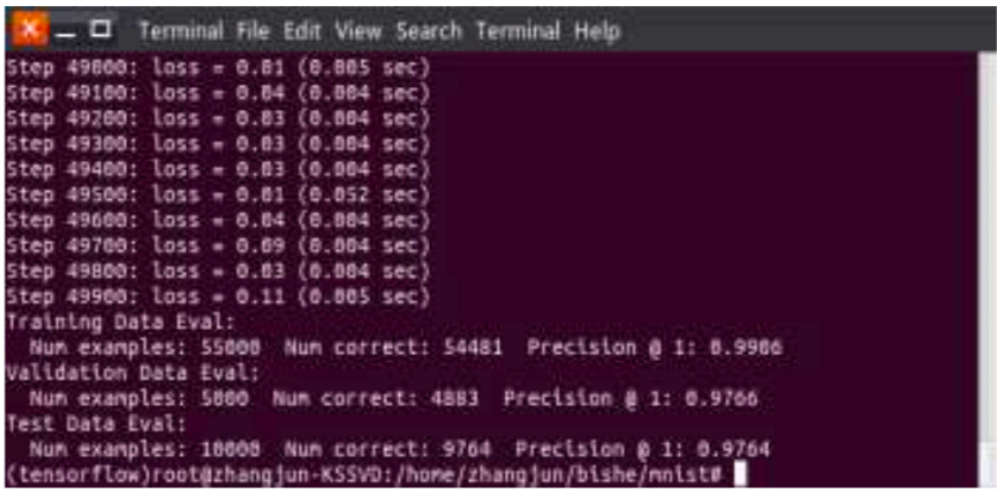


找到所有预测正确标签的数量，得出模型预测的正确率

```
correct = tf.nn.in_top_k(logits, labels, 1)

return tf.reduce_sum(tf.cast(correct,tf.int 32))
```

通过训练模型，最终得到下图所示的结果:训练集的正 确率为99.06%;验证集数据正确率为 97.66%;测试集数据正确率为:97.64%。



总结

在本学期的人工智能课程学习中，首先我们接触并学会了制作一个智能小车，了解到了计算机硬件上的应用，同时也加强了动手能力。在此，对于小车功能拓展部分，逐步思考了如何将硬件与代码结合起来进而决定对字符识别做进一步的探索。

利用课堂上学习的对与 MINIST 数据集的多种处理方式，了解到逻辑回归、SVM 向量机、CNN 均可以对其做识别分类，经过查找资料和实际实践后，这里提出了一种较为高效的识别算法：BP 神经网络。

通过本学期的学习，初步了解了人工智能的应用与操作方法。目前实验还有很多不足，小车的运行也不稳定，今后还可以根据多样的传感器拓展更多的功能。

参考文献:

[1] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large- scale machine learning on heterogeneous systems, 2015[J]. Software available from tensorflow. org.

[2] 罗亮，陈红，卢解卿. 谈如何用Python控制Arduino[J]. 中国信息技术教育，2016(8):65-67.

[3] 张斌,赵玮烨,李积宪.基于BP神经网络的手写字符识别系统 [J].兰州交通大学学报:自然科学版, 2007,26(1).

[4] Goldsborough P. A Tour of TensorFlow[J]. 2016.