人工智能期末论文

一、课程学习

循环神经网络：RNNs学习

0. 什么是RNNs

RNNs的目的是用来处理序列数据的。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。而这种神经网络对于要预测句子的下一单词之类的序列数据问题，却是无能为力的，因为它要求的一个句子前后单词并不是独立的。

RNNs之所以称为循环神经网络，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆，并应用于当前输出的计算中，即隐藏层之间的节点不再是无连接而是有连接的。并且隐藏层的输入不禁包括输入层的输出，还包括上一个时刻隐藏层的输出。并且输出层的结果，还会通过反向传播来影响隐藏层。

1. 什么是神经元记忆：

对于一首熟悉的歌词，直接跳跃到第二小节的歌词比较困难，一般正常的情况下，我们是将其作为一个序列去记忆的。我们的记忆其实是和链表的结构是类似的，当我们试图从直接记忆起中间的部分的时候，便是从这首歌的开头所在的神经元作为索引开始寻找的。

记忆这样去存储是很有效的，因为这样可以通过脑神经网络很好的找到相似的属性、优势。

而我们想要去训练一个神经网络，去预测下一帧的情况，这时候，知道上一帧的情况会对预测很有帮助，而这样的序列数据就是我们为什么要去搭建一个循环神经网络。神经网络如何去记住它之前的时间它所看到的元素。

神经网络有隐藏层，一般，隐藏层状态只和输入数据有关，所以一个神经网络的信息流就如下面所示:

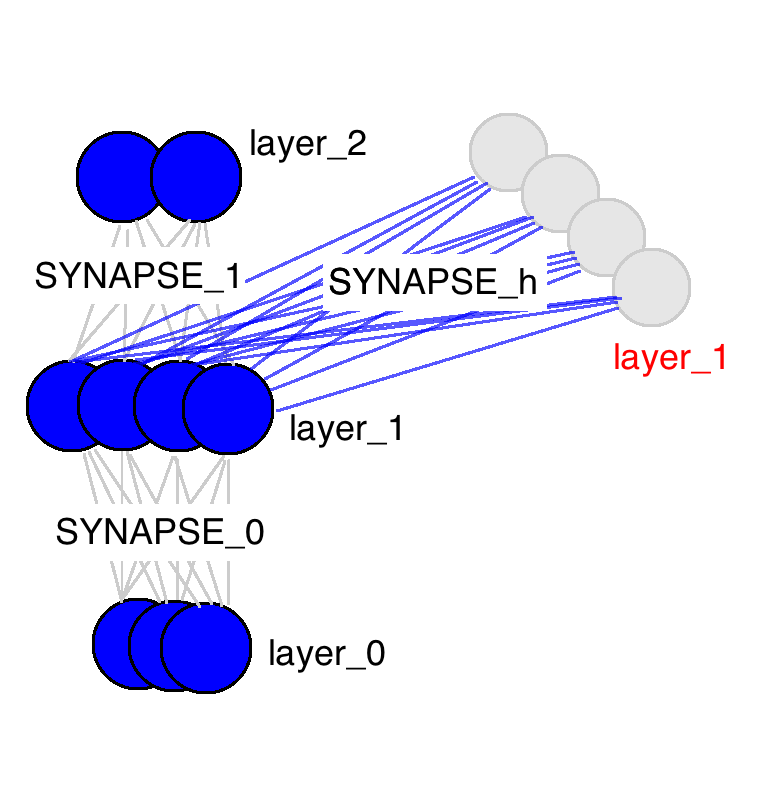
Input –> hidden -> output

这是一种确定的输入产生确定的隐藏层，确定的隐藏层产生确定的输出层。这是一种封闭系统，但是记忆改变了这种模式。记忆意味着是隐藏层是，当前时刻的输入与隐藏层前一时刻的一种组合。

( Input + prev\_hidden ) -> hidden -> output

2. RNN – 神经网络记忆

输入数据决定了我们神经网络的输入层，每行输入数据都被用来产生隐藏层，然后用每个隐藏层来生成输出层（假设只有一层隐含层）。记忆意味着隐含层是输入与上一次隐含层的组合，而组合的方式，就是用一个矩阵来定义了之前隐藏层与当前的关系。



SYNAPSE\_0把输入数据传播到隐藏层，SYNAPSE\_1把隐藏层数据传播到输出层。而新的矩阵SYNAPSE\_h是每一次循环所产生的，把隐藏层(layer\_1)传播到下一个时间点的隐藏层(layer\_1)。

正如同记忆一样，神经网络的隐藏层也是会有容量的。当一个隐藏层每个节点都由输入和前面的隐藏层所影响时，再进行下一步循环的时候，便会不得不选择某一个节点去替代。同样神经网络学习中，也需要做到忘记无关的记忆，并且记住重要的记忆。

3. 基于时间的反向传播：

网络通过正向的全部传输，再由输出层从末尾到开头的反向传播所有的导数值。

4. 一个实例代码：

通过RNNs，在给定一个开头，来模仿周杰伦的风格，续写这首歌。

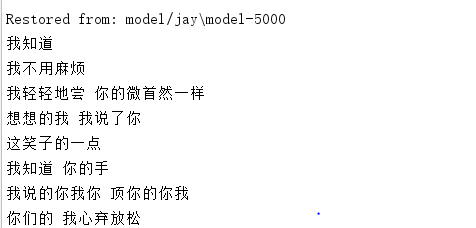
核心代码如下：

训练代码：

**def** train(self, batch\_generator, max\_steps, save\_path, save\_every\_n, log\_every\_n):  
 self.session = tf.Session()  
 **with** self.session **as** sess:  
 sess.run(tf.global\_variables\_initializer())  
 *# Train network* step = 0  
 new\_state = sess.run(self.initial\_state)  
 **for** x, y **in** batch\_generator:  
 step += 1  
 start = time.time()  
 feed = {self.inputs: x,  
 self.targets: y,  
 self.keep\_prob: self.train\_keep\_prob,  
 self.initial\_state: new\_state}  
 batch\_loss, new\_state, \_ = sess.run([self.loss,  
 self.final\_state,  
 self.optimizer],  
 feed\_dict=feed)  
  
 end = time.time()  
 *# control the print lines* **if** step % log\_every\_n == 0:  
 print(**'step: {}/{}... '**.format(step, max\_steps),  
 **'loss: {:.4f}... '**.format(batch\_loss),  
 **'{:.4f} sec/batch'**.format((end - start)))  
 **if** (step % save\_every\_n == 0):  
 self.saver.save(sess, os.path.join(save\_path, **'model'**), global\_step=step)  
 **if** step >= max\_steps:  
 **break** self.saver.save(sess, os.path.join(save\_path, **'model'**), global\_step=step)

将每一步的loss输出出来

训练输出的结果如下:



可以看到，虽然有一些语句还是不通畅，但是总体来说，还是有着周杰伦歌词的风格。而从结果可以看到，在语句不通畅的地方，通常都是你、我这些出现频率较高的词上面。如果将数据集处理之后，便能够减少这种不通畅的地方。

二、课程实践：

在课程装小车的过程中，一共进行了三次的装小车活动：

第一次的目的是组装小车，并且完成小车的一些基本功能，我所做的工作，便是将小车的代码写好并调试，最后上传到小车的Arduino板上。

要实现的功能如下：

Loop() 模块 ： 实现不断的重复，如果没有蓝牙信号输入，便自动驾驶，如果蓝牙信号输入，便成为手动控制的小车。

Forward() 模块 ： 前进

Backward()模块 ： 后退

Stopcar() 模块 ： 停车

Left() 模块 ： 向左

Right() 模块 ： 向右

Drive() 模块 ： 将蓝牙传输的信号转换为对应的 Forward、Backward、Stopcar、Left 、 right 模块。

Getdistance() 模块 ： 将传感器感受到的前方障碍物离小车的距离返回

UltrasonicCar() 模块 : 自动避障模块，在小车的前方障碍物的距离小于20cm时，便使得小车向右转向，否则继续前进。

Randomright() 模块:：随机右转模块，在小车遇到前方有障碍物的时候向右转向时，右转时间为一个随机数，0.05 – 1秒之间。

第二次的目的是在小车的基础功能实现的情况下，再完成语音识别和图像识别，我所完成的工作是，开发一个蓝牙串口 + 语音识别功能的APP

具体情况如下：

首先先是调用了讯飞SDK，再在其基础上添加蓝牙模块。

模块如下：

SearchBluetoothDevice() 模块 ： 搜索蓝牙设备

CheckBluetoothSupport() 模块 ： 检查是否支持蓝牙

InitView() 模块 ： 搜索蓝牙设备，搜索之前先断开当前连接，并且清空存储蓝牙设备的arrayList，再搜索结束后，把结果输出。

InitLiserner() 模块 ： 初始化监听器模块

RecognizerListener() 模块 ： 实现语音识别的模块，通过将当前录音的语音，通过网络发送给讯飞数据块，再由该数据块对该录音进行分析处理，得到最后的结果。

OnResume()模块 ： 开放统计，移动数据统计分析

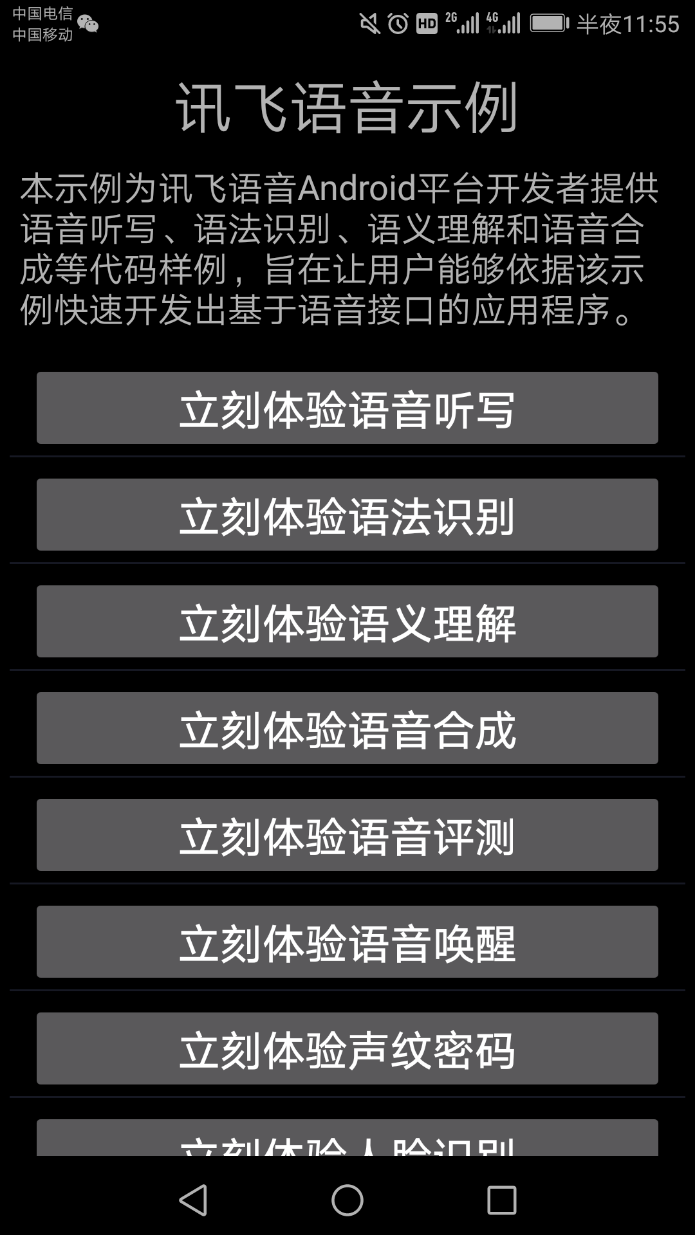
OnPause()模块 ： 开放统计 ， 移动数据统计分析

APP的logo是通过图画本绘制而成



G9的含义便代表着Group 9

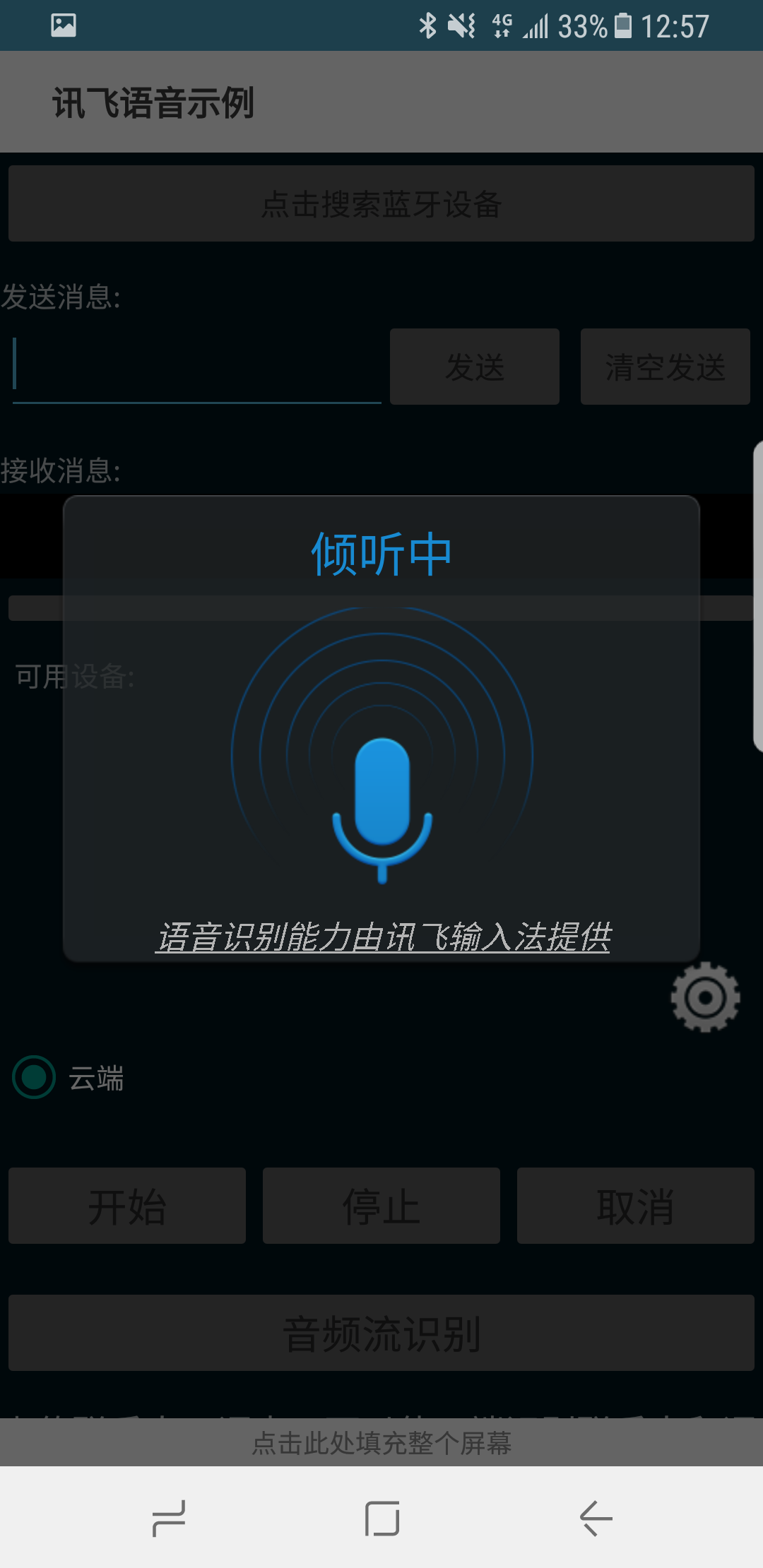
APP的界面如下：



搜索蓝牙功能：



语音识别功能：



第三次的目标是完成小车图像识别功能和机械臂模块

要达成的目标是通过发送语音给小车要抓取某物，小车通过语音识别出要抓取的名词，然后通过图像识别去找到它，再由机械臂将它抓取。

三、课程总结

在这门课上，学会了Arduino开发，增强了自身的动手能力，通过也增加了自己的创造能力，比如，想到通过开发一个APP来实现小车的语音识别功能，而且在这门课上，还学会了一些神经网络的算法，也对于如何使用tensorflow编程有了一定的了解。