学生姓名：宋 锐

学号：2016202141

2018年12月27日

**人工智能导论课程报告**

## 人工智能课程报告

#### 1.引言

人工智能（AI），研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。70年代，人工智能技术面临衰落，2016年Google AlphaGO战胜韩国围棋棋王李世石，人工智能一度成为耳熟能详的热词。当前人工智能产业发展主要是基于深度学习算法，在数据量和计算能力的基础上实现了大规模计算。人工智能是对人的意识、思维的信息过程的模拟。在某些危险性工作中，已经选择并应用人工智能代替人类做此类高危工作。在中国，人工智能的计算机视觉领域拥有最多创业公司且目前我国计算机视觉水平达到了全球领先水平，其次则是服务机器人领域。

展望未来，人工智能一方面赋予机器反应和适应能力优化产出，通过物联网、机器人等技术的结合构造出一个整合的信息物理世界；另一方面人工智能可能成为一些社会核心挑战的有利工具。例如医疗领域，人工智能可以提高我们分析人类基因组；环保领域，人工智能能够分析其后特征并大规模降低能耗等。

#### 2.学习成果概述

通过一个学期人工智能导论学习及实践，我主要收获了以下知识：

1. 学习了python环境的搭建，在搭建环境过程中遇到很多配置问题，通过解决这些问题提高了自己对python环境配置的了解程度。
2. 学习了基于arduino板的小车安装、了解了小车的超声波模块、蓝牙模块等，通过arduino IDE和计算机向小车烧制例程代码，实现了小车随机游走。
3. 系统学习了图像识别、卷积神经网络部分内容，学习理解了CascadeClassifier原理、face\_recogintion原理、openCV包等。
4. 提高了自己筛选及查询有价值的资料的能力，例如通过Github、Google等平台学习了arduino小车相关知识，学会了利用开源app（蓝牙串口、“迹”app、IP摄像头）与小车进行交互，学习了机器学习的相关内容。
5. 系统梳理学习了tensorflow的模型训练、基于openCV的CNN算法、HOG+SVW算法等图像识别的相关知识。

#### 3.小组作业

###### 3.1小组作业实现情况

本学期中，在人工智能课程实践中，共完成了三次展示，展示成果分别为以下：

（1）第一次展示成果

1. 完成arduino小车搭建；
2. 通过arduino IDE与计算机的交互实现了红外线循迹，超声波避障及红外线避障、通过蓝牙串口实现了蓝牙遥控，及点击手机端蓝牙串口按钮设置实现直走、转弯、旋转等。
3. 通过“迹”APP及蓝牙串口的使用实现了追踪某种特定颜色的物体，可以做到 直线跟踪、曲线跟踪、当指定物体消失之后，原地按照目标物体最后一次在屏幕中出现的位置选择方向旋转寻找目标。

（2）第二次展示成果

1. 实现图像识别。通过openCV自带的haar人脸特征分离器进行人脸检测，再基于face\_recognition用cnn算法实现静态人脸识及动态人脸识别。具体表现为识别出指定人脸则小车前进，否则停止。
2. 实现语音控制小车。基于DTW和GMM-HMM识别“前”、“后”、“右”、“停”五个指令，通过电脑蓝牙与小车蓝牙连接，从而实现语音控制小车运动。
3. 第三次展示成果
4. 实现小车自动驾驶。基于openCV使用CNN算法实现小车自动驾驶，在手机上搭建IP摄像头APP，通过IP摄像头APP为小车增加眼睛，电脑通过蓝牙控制小车，实现自动识别搭建道路并避障。
5. 通过CNN实现识别交通标志，最后实现小车在有标志的搭建道路上实现自动驾驶。

###### 3.2小组作业参与情况

1. 第一次实验中，参与小车的组建安装，了解了小车构造，通过计算机端口及小车蓝牙的交互及例程代码实现了小车随机游走，包括直走，超声波避障、红外线循迹。最后参与PPT制作及视频处理。
2. 第二次实验中，参与小车图像识别的实现，了解了python自带的大量功能强大的包，例如face-recognition等，了解了卷积神经网络在深度学习的运用。尝试了在手机端搭建Tensorflow环境直接运行代码控制小车，尝试了“警猫眼”、“Android stdio”等软件，最终由于搭建过于复杂而放弃。
3. 第三次实验中，协助参与小车自动驾驶的实现，主要参与测试集的采集工作。

###### 3.3小组作业遇到的问题以及收获

1. 第一次实验中，第一次进行超声波避障测试时，发现小车很难实现完美避障，一开始是障碍物过小，低于小车超声波模块的安装位置，从而无法实现完美避障；替换为较大障碍物之后，小车实现完美避障，但是若小车前方障碍物过于密集，小车不停旋转超声波模块，无法退出当前位置。

提高了探索错误原因及调试程序的能力。

1. 第二次实验中，通过ip摄像头从小车端向电脑端传回图像资料进行处理时，由于网络原因导致小车控制不明显，最终选择直接通过电脑端摄像头收集图像资料进行识别，通过蓝牙发送命令控制小车前进或者停止。由于时间原因及手机端tensorflow环境搭建较 复杂，最终未实现手机端直接运行代码控制小车。

提高了采用多种方法解决问题的能力，了解到手机端搭建tensorflow的可能，对于以后的学习积累了经验。了解了“迹” APP的工作原理，学会了使用该APP进行小车目标跟随实验。

1. 第三次实验中，收集到的训练集训练之后准确率低，召回率低，使用训练后的模型进行尝试实验时，小车无法识别已经搭建的“跑道”，表现效果为直走或者原地旋转寻找“跑道”。本来尝试编写APP实现小车控制，最终由于android端APP由Java语言进行编写，队友对于java了解甚少，最终放弃放弃编写APP。网络延迟严重，导致小车行进比较缓慢。训练集照片数量不足，导致训练的模型准确度不高。

学习了基于openCV的CNN算法，学习了模型的训练集收集及模型训练。

#### 课后代码实现

###### 4.1理论知识学习

在学习中，与队友完成课程实践之外，尝试学习了基于openCV的HOG特征算法进行图像识别，虽然最终由于实现按效果不理想未使用该程序代码，但将学习过程记录如下。HOG（[Histogram of Oriented Gridients](http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf" \t "https://blog.csdn.net/hujingshuang/article/details/47337707/_blank)）特征算法是一种解决人体目标检测的图像描述子，是一种用于表征图像局部梯度方向和梯度强度分布特性的描述符。

HOG特征检测算法主要由以下步骤：图像灰度化、图像归一化、梯度计算、计算梯度直方图、重叠块直方图归一化。下面将详细介绍执行过程及代码实现。

**步骤一：图像灰度化。**

在图像采集的过程中，可能会由于背景的多样性导致采集的人脸效果不好，导致出现人脸错检或者遗漏的情况，此步骤就是对采集到的人脸进行图像预处理，处理环境造成的光线不合适的问题，利用图像灰度化处理将图像进行灰度处理，避免光线太强或者太弱的影响；

**步骤二：图像归一化**。

利用Gamma校正处理图像光线强度不均匀的情况，将图像整体亮度提高或降低。

**步骤三：梯度计算。**

对图像灰度化及图像归一化后的图像计算每个像素的梯度及梯度方向，分别在水平及垂直方向进行计算，捕获轮廓信息。

**步骤四：计算梯度直方图。**

将图像划分为若干个单元（cells）,一般默认8x8=64个像素为一个cell，且临近的cells之间不重叠，在每个cell内统计梯度方向直方图，即不同梯度的个数，形成每个cell的描述符。

**步骤五：重叠块直方图归一化。**

将图像内所有有重叠部分block的直方图进行对比度归一化，也就是将所有block内的直方图向量进行组合，得到目标图像的HOG特征descriptor，结果即为最终分类的特征向量。

###### 4.2 代码实现

1. **数据准备。**

#图像灰度化

img = cv2.imread('person.png', cv2.IMREAD\_GRAYSCALE)

#令GAMMA=0.5进行校正

img = np.sqrt(img / float(np.max(img)))

1. **计算每个像素的梯度。**

def global\_gradient(self):  
 gradient\_values\_x = cv2.Sobel(self.img, cv2.CV\_64F, 1, 0, ksize=5)  
 gradient\_values\_y = cv2.Sobel(self.img, cv2.CV\_64F, 0, 1, ksize=5)  
 gradient\_magnitude=cv2.addWeighted(gradient\_values\_x,0.5, gradient\_values\_y, 0.5, 0) gradient\_angle=cv2.phase(gradient\_values\_x,gradient\_values\_y,

angleInDegrees=True)

return gradient\_magnitude, gradient\_angle

1. **计算每个cell内的梯度方向直方图。**

def cell\_gradient(self, cell\_magnitude, cell\_angle):  
 orientation\_centers = [0] \* self.bin\_size  
 for i in range(cell\_magnitude.shape[0]):  
 for j in range(cell\_magnitude.shape[1]):  
 gradient\_strength = cell\_magnitude[i][j]  
 gradient\_angle = cell\_angle[i][j]

min\_angle,max\_angle,mod=self.get\_closest\_bins(gradient\_angle)  
orientation\_centers[min\_angle] += (gradient\_strength \* (1 - (mod / self.angle\_unit)))  
orientation\_centers[max\_angle] += (gradient\_strength \* (mod / self.angle\_unit))  
 return orientation\_centers

1. **计算cell梯度方向直方图，计算block的梯度信息，将cell组合为block，进行块内直方图归一化。**

def extract(self):  
 height, width = self.img.shape #计算图像的HOG描述符，以及HOG-image特征图  
 gradient\_magnitude, gradient\_angle = self.global\_gradient()  
 gradient\_magnitude = abs(gradient\_magnitude)

#计算输入图像的每个cell单元的梯度直方图，形成每个cell的descriptor  
 cell\_gradient\_vector = np.zeros((height / self.cell\_size, width / self.cell\_size, self.bin\_size))  
 for i in range(cell\_gradient\_vector.shape[0]):  
 for j in range(cell\_gradient\_vector.shape[1]):

#计算第[i][j]个cell的特征向量  
 cell\_magnitude=gradient\_magnitude[i \* self.cell\_size:(i +1) \* self.cell\_size,j \* self.cell\_size:(j + 1) \* self.cell\_size]  
 cell\_angle=gradient\_angle[i\*self.cell\_size:(i+1)\*

self.cell\_size,j \* self.cell\_size:(j + 1) \* self.cell\_size]  
 cell\_gradient\_vector[i][j]=self.cell\_gradient(cell\_magnitude, cell\_angle)

#得到每个cell的梯度方向直方图，得到特征图 hog\_image=self.render\_gradient(np.zeros([height,width]),cell\_gradient\_vector)  
 hog\_vector = [] #最终分类的特征向量   
 for i in range(cell\_gradient\_vector.shape[0] - 1):  
 for j in range(cell\_gradient\_vector.shape[1] - 1):  
 block\_vector = []  
 block\_vector.extend(cell\_gradient\_vector[i][j])  
 block\_vector.extend(cell\_gradient\_vector[i][j + 1])  
 block\_vector.extend(cell\_gradient\_vector[i + 1][j])  
 block\_vector.extend(cell\_gradient\_vector[i + 1][j + 1])

#块内归一化梯度直方图，使光照强度均匀，集体提高或降低亮度  
 mag = lambda vector: math.sqrt(sum(i \*\* 2 for i in vector))  
 magnitude = mag(block\_vector)  
 if magnitude != 0:  
 normalize=lambda block\_vector, magnitude: [element / magnitude for element in block\_vector]  
 block\_vector = normalize(block\_vector, magnitude)  
 hog\_vector.append(block\_vector)  
 return hog\_vector, hog\_image

HOG特征算法+SVW分类器广泛用于图像识别，上述过程实现了HOG特征算法，在学习HOG特征算法时，了解到多种人脸检测算法：例如例如openCV自带的haar人脸检测算法、HOG算法、以及特征脸方法（Eigenface）等。

#### 学习内容总结

笼统来说，机器学习是模仿人识别事物的过程。从大量数据中提取规则或模式将数据转换成信息，主要的学习方法有归纳学习法和分析学习法。进行数据预处理后，形成特征，然后根据特征创建特定模型。

机器学习中存在多种分类算法，例如决策树、CNN等。

1. 决策树

决策树的学习过程主要有以下三步：

* 特征选择：从训练数据中的特征中选择某个特征作为当前节点的分裂标准。
* 决策树生成：根据选择的特征评估标准，从上至下递归地生成子节点，知道数据集不可分则停止决策树的生长。
* 剪枝：决策树容易过拟合，此时需要剪枝，缩小树结构规模、缓解过拟合问题。

1. CNN（Convolutional Neural Network）

* 输入层：对输入采取去均值、归一化或PCA/SVD降维方式进行预处理。
* 卷积层：卷积层使用“卷积核”进行局部感知，减少了模型的计算参数。
* 激励层：对卷积层的输出结果进行非线性映射，而激励函数的选择是多样性的，典型激励函数有Sigmoid函数、Tanh函数等。
* 池化层：主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。
* 全连接FC层：经过前面若干次卷积+激励+池化后，模型会将学到的一个高质量的特征图片全连接层。

1. 机器学习总结。

机器学习的内容繁多而冗杂，而对于例如上述经常提到的HOG算法、haar特征检测、CNN等经典算法，网络或者书本、甚至课堂上都有讲解及源码介绍，在学习中应当做到一方面知悉其原理，另一方面学会调试及使用相关代码实现人工智能相应功能。