

计算机视觉部分文档

- 1 问题概述
- 2 问题分析
- 3 模块功能概述
- 4 模块主体代码编写
- 5 卷积神经网络的实现
 - 5.1 收集数据并搭建数据集类
 - 5.2 搭建卷积神经网络
 - 5.3 卷积神经网络训练的预备工作
 - 5.4 卷积神经网络训练结果
 - 5.5 对卷积神经网络的改进
- 6 假设检验

1 问题概述

视觉部分的需求是从摄像机中读取图像，然后判断该图像中是否有人物出现。该人物必须面朝机器人且注视机器人，否则将被视作仅从机器人面前经过而非想要和机器人交互。

2 问题分析

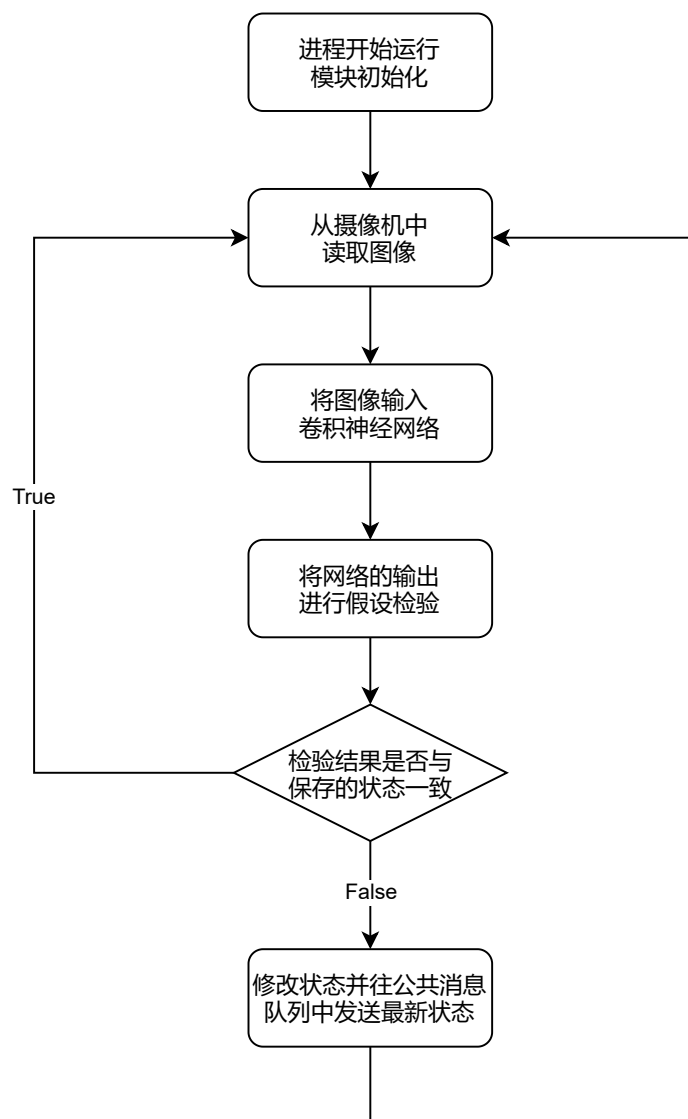
本问题的核心是如何判断图像中的人物面朝机器人并且注视机器人，经过考虑本部分使用卷积神经网络来进行实现。

3 模块功能概述

本模块运行在一个独立的进程中。模块的状态置为没有人出现并注视。

然后模块会从机器人的摄像机中读取图像，经由训练好的卷积神经网络判断图像中是否有人物出现并注视机器人。由于卷积神经网络并非能准确判断，所以引入假设检验来提高判断准确率。将判断的结果与现有的状态比较，若相同则不做任何处理，若不同则会修改状态并往公共消息队列中发送最新的状态。重复上述过程。

运行在其他进程中的其他模块会去读取公共消息队列并根据公共消息队列中的最新状态采取不同的行动。



4 模块主体代码编写

5 卷积神经网络的实现

5.1 收集数据并搭建数据集类

为了使网络有良好的表现，一共收集**16,461**张正视摄像头的人像图像用作识别正确的部分和**5,468**张室内场景图像用作识别错误的部分。

数据集结构为：

- dataset
 - True
 - img1

- `img2`
- `...`
- `False`
- `img1`
- `img2`
- `...`

通过继承Pytorch中的 `Dataset` 类构建自己的 `dataset` 类，命名为 `MyDataset`。该数据集类把所有数据中的前80%划作训练集，后20%划作测试集。

同时 `MyDataset` 中还有 `train` 和 `transform` 两个参数，前者负责设置该 `MyDataset` 实例是作为训练集还是测试集，后者负责对从 `MyDataset` 实例中取出的图像做对应的变换。

```

1  class MyDataset(Dataset):
2      def __init__(self, root_path, train, transform):
3          super().__init__()
4          self.root_path = root_path           # 保存根目录
5          self.label_path = os.listdir(self.root_path) # 获得所有标签
6          self.imgAndLabel = []                # 用于存放图像地址和对应标签
7          self.transform = transform           # 保存对图像进行的变换
8
9          if train:
10             # 如果用作训练集
11             for label in self.label_path:
12                 path = os.path.join(self.root_path, label)
13                 img_path = os.listdir(path)
14
15                 # 将数据集的前80%放在imgAndLabel中用作训练集
16                 total_img_size = round(len(img_path)*0.8)
17                 for i in range(total_img_size):
18                     self.imgAndLabel.append(

```

```
19         (os.path.join(self.root_path,
20                         label, img_path[i]), label)
21     )
22     else:
23         # 如果用作测试集
24         for label in self.label_path:
25             path = os.path.join(self.root_path, label)
26             img_path = os.listdir(path)
27
28             # 将数据集的后20%放在imgAndLabel中用作测试集
29             total_img_size = round(len(img_path)*0.8)
30             for i in range(total_img_size, len(img_path)):
31                 self.imgAndLabel.append(
32                     (os.path.join(self.root_path,
33                                 label, img_path[i]), label)
34                 )
35
36     def __getitem__(self, index):
37         # 获得图像路径和标签
38         img_path, label = self.imgAndLabel[index]
39
40         # 读取图像
41         img = cv.imread(img_path)
42
43         # 如果有变换就对图像应用变换
44         if self.transform:
45             img = self.transform(img)
46
47         # 将标签转换为对应的tensor类型的数据方便使用
48         label_tensor = 0
49         for i in range(len(self.label_path)):
50             if label == self.label_path[i]:
```

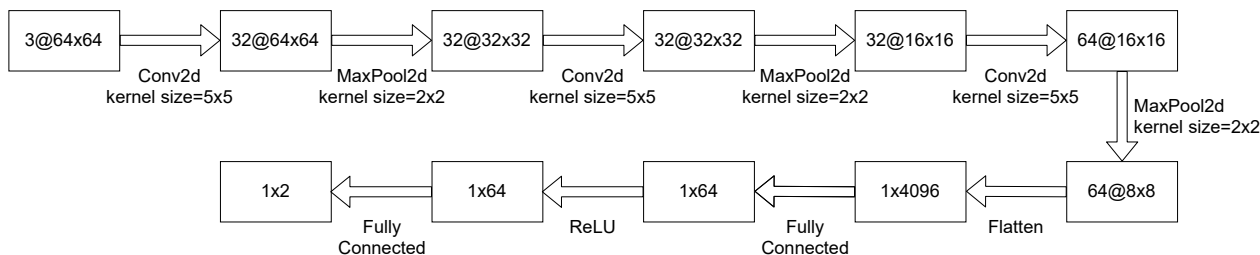
```
51         label_tensor = torch.tensor(i)
52         break
53
54     return img, label_tensor
55
56 def __len__(self):
57     return len(self.imgAndLabel)
```

5.2 搭建卷积神经网络

因为此次卷积神经网络的任务是一个二分类问题，任务本身比较简单，所以不设置较深较宽的网络结构。

网络结构采用如下设置：

1. 卷积层，输入通道数为3，输出通道数为32，卷积核大小为5x5，边缘采用0填充。
2. 最大值池化层，池化核为2x2。
3. 卷积层，输入通道数为32，输出通道数为32，卷积核大小为5x5，边缘采用0填充。
4. 最大值池化层，池化核为2x2。
5. 卷积层，输入通道数为32，输出通道数为64，卷积核大小为5x5，边缘采用0填充。
6. 最大值池化层，池化核为2x2。
7. Flatten层，将向量展平。
8. 全连接层，输入为4096维向量，输出为64维向量。
9. ReLU层，用作上层全连接层的激活函数。
10. 全连接层，输入为64维向量，输出为2维向量。



其中卷积层的作用是提取图像中的各种有用的数据，通过多层卷积层后能够提取出图像中高维的信息。相比于直接使用不经任何处理的图像进行学习的网络，卷积神经网络能大幅提高网络训练的效率并减小网络的规模。

池化层的作用也是抽取数据，缩小数据规模的同时保留原数据中的主要特征。**Pytorch**中的池化层同时兼顾取最值和降采样的效果，取最值可以保留原数据中的主要特征，而降采样便于后面的卷积层提取出更高维的数据特征。

Flatten层的作用是将经过三层卷积层和三层池化层后的**64x8x8**的向量展开变成**4096**维向量。

全连接层相当于神经网络中的整合函数，负责对数据进行加权求和并加上偏置值。

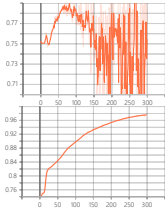
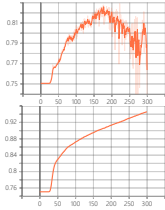
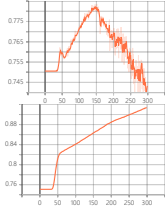
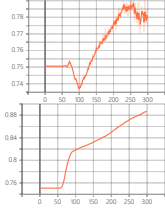
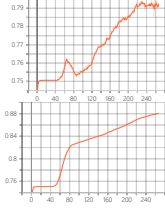
ReLU层则是神经网络中的激活函数，其函数形式如下所示

$$f(x) = \max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

5.3 卷积神经网络训练的预备工作

由于任务是二分类，所以训练使用交叉熵作为损失函数，使用随机梯度下降作为优化器，使用**DataLoader**加载数据集。通过对**DataLoader**的**batch size**的设置，实际上的优化算法应该等价于小批量梯度下降算法。

首先进行超参数的设置。先设置**DataLoader**的**batch size**，让学习率保持**0.0001**，**batch size**从**32**开始，每次加上**32**，观察网络的拟合情况。

batch size	训练结果
32	
64	
96	
128	
160	
192	
224	
256	

由于时间原因，每个batch size的实验只能进行一次，实验所得数据有很大的偶然性和不稳定性。但也能从这些数据中看出这样的趋势：batch size越大，小批量梯度下降算法就越不容易过拟合。当batch size较小时，即使学习率设置为0.0001，网络权值在训练到60轮左右的时候就会过拟合；而batch size增加后，300轮的训练也不能看到网络权值的过拟合。

5.4 卷积神经网络训练结果

5.5 对卷积神经网络的改进

6 假设检验