

产品缺陷检测

实验报告

关钰千 |10142510128| 人工智能基础 | 2018年10月10日

目录

[一、 实验目的 2](#_Toc528500072)

[二、 实验内容 2](#_Toc528500073)

[1. 输入输出定义 2](#_Toc528500074)

[2. 判定规则 2](#_Toc528500075)

[三、 设计思想 2](#_Toc528500076)

[1. 检测思路 2](#_Toc528500077)

[2. 特征点提取 2](#_Toc528500078)

[3. 丰富数据集（可选） 2](#_Toc528500079)

[4. 判定思路 2](#_Toc528500080)

[四、 实验环境 2](#_Toc528500081)

[五、 实验过程 3](#_Toc528500082)

[1. 工程目录 3](#_Toc528500083)

[2. 图像分类 3](#_Toc528500084)

[3. 标签处理 3](#_Toc528500085)

[4. 常数类实现（constant.py） 3](#_Toc528500086)

[5. 图像预处理( image\_preprocessing.py) 3](#_Toc528500087)

[6. 神经网络训练（ gpu\_train.py  ） 3](#_Toc528500088)

[7. 神经网络提升 4](#_Toc528500089)

[8. 产品检测（predict.py） 4](#_Toc528500090)

[六、 实验结果（result\_images ） 4](#_Toc528500091)

[1. 图片结果输出 4](#_Toc528500092)

[2. 文本结果输出 5](#_Toc528500093)

[七、 实验总结 7](#_Toc528500094)

[1. 关于方法选择 7](#_Toc528500095)

[2. 数据集大小的思考 7](#_Toc528500096)

[3. 深度学习入门的思考 7](#_Toc528500097)

[4. 深度学习的基础知识掌握 7](#_Toc528500098)

# 实验目的

* 掌握人工智能相关基础知识
* 掌握人工智能中图像识别的基础知识
* 掌握机器学习编程方法

# 实验内容

## 输入输出定义

输入规则：给定一组产品焊接图片

输出规则：区分输入中的合格和不合格产品，并指出不合格产品中的缺陷点

## 判定规则

焊接成功：当几个焊点都焊接在两个工件的连接处时，说明该焊接成功

产品合格：一张输入的图片中的所有焊接都成功则称该产品合格

产品不合格：一张输入产品图片中出现一次以上焊接不成功则称该产品不合格

# 设计思想

## 检测思路

把产品图片中的所有焊接分离，对每个焊接进行检测，最后根据每个焊接的判定结果输出。

## 特征点提取

由于产品照片机位相似，是否合格的特征在一定的固定区域以内，故用简单的人工特征点放大，把产品图片的特征的区域分离出来进行检测，减少不相关干扰。

## 丰富数据集（可选）

由于实验给的数据集过少，所以这里对经过预处理的图像进行旋转、镜像等操作得到新的数据集，这可能会让训练结果更加好，但是最后使用原数据集的操作就已经达到100%的正确率（检测集较少的原因），没有更多数据表明图像处理丰富的数据集可以让训练结果更好，所以最后的实验就没有使用。

## 判定思路

利用深度学习的方法，搭建深度学习框架，把根据上述思路进行预处理的图像作为数据集，训练出一个用于产品缺陷检测的神经网络。

# 实验环境

* 运行环境：Python
* 深度学习框架：PyTorch
* 图像处理库：PIL，Skimage
* 开发环境：Windows 10，GTX 1063，PyCharm，intel i5 7500

# 实验过程

## 工程目录

1. ├─ constant.py                   // 超参数定义为常量
2. ├─ gpu\_train.py                  // cnn训练神经网络
3. ├─ image\_preprocessing.py        // 图像预处理
4. ├─ net\_last.pkl                  // 最后一次训练的结果
5. ├─ net\_no\_validation\_acc1.0.pkl  // 无测试集训练神经网络，acc100%
6. ├─ net\_validation0.3\_acc1.0.pkl  // 30%测试集训练神经网络，acc100%
7. ├─ predict.py                   // 预测图像路径中的图像，并输出结果
8. ├─ result\_images // 图像结果输出在此文件夹下
9. └─ original\_images              // 存储图片文件的
10. ├─check\_images            // 预测集图片路径
11. ├─failure\_images              // 训练集失败图片路径
12. └─success\_images          // 训练集成功图片路径

## 图像分类

图片存于original\_images文件夹内，其中check\_images中放置我们的预测集，failure\_images中放置不合格的产品图片，success\_images中放置合格的产品图片。

## 标签处理

图片按照标签命名，如图片中焊接分别为成功-失败-失败，则该图片命名为100 (x).jpg（x是整数，防止重复命名）。

## 常数类实现（constant.py）

鉴于编程的良好规范，常量都定义在constant.py中，并且进行大写和不可修改的规定。

## 图像预处理( image\_preprocessing.py)

* 根据图片中的三个焊接大致区域，将图片分出三份
* 根据特征点位于焊接的大致区域，将焊接图片焊接点大致区域裁剪出来
* 采用 镜像\*180度翻转 丰富数据集

## 神经网络训练（ gpu\_train.py  ）

输入数据处理：把数据打乱，分为70%的训练集和30%的测试集，由于数据集较少，所以预测集没有意义，就没有分离出预测集。

起初采用的是TensorFlow的MLP进行建模训练，最后得出的效果并不是很好，之后用了Pytorch的cnn进行建模训练。训练时用测试集得出的超参数如下代码

1. EPOCH = 30# train the training data n times
2. BATCH\_SIZE = 50
3. LR = 0.0001              # learning rate
4. if\_use\_gpu = 1
6. **class** CNN(nn.Module):
7. **def** \_\_init\_\_(self):
8. super(CNN, self).\_\_init\_\_()
9. self.conv1 = nn.Sequential(         # input shape (1, 28, 28)
10. nn.Conv2d(
11. in\_channels=1,              # input height
12. out\_channels=16,            # n\_filters
13. kernel\_size=5,              # filter size
14. stride=1,                   # filter movement/step
15. padding=2,
16. ),                              # output shape (16, 28, 28)
17. nn.ReLU(),                      # activation
18. nn.MaxPool2d(kernel\_size=2),
19. nn.Dropout(0.2)
20. )
21. self.conv2 = nn.Sequential(         # input shape (16, 14, 14)
22. nn.Conv2d(16, 32, 5, 1, 2),     # output shape (32, 14, 14)
23. nn.ReLU(),                      # activation
24. nn.MaxPool2d(2),                # output shape (32, 7, 7)
25. nn.Dropout(0.5),
26. )
27. self.out = nn.Linear(32\*int(constant.WIDTH\_PX/4)\*int(constant.HEIGHT\_PX/4), 2)
29. **def** forward(self, x):
30. x = self.conv1(x)
31. x = self.conv2(x)
32. x = x.view(x.size(0), -1)
33. output = self.out(x)
34. **return** output, x    # return x for visualization

## 神经网络提升

使用Pytorch的cnn模型，最后已经让验证集的准确率达到100%。为了让模型进一步提升，让验证集的那30%的数据也参与训练。由于最后的出来的结果始终都是100%，数据集太少，所以之前想扩展数据集的方式也成了可选方案，并未采用，因为无法验证扩展数据集是否可以训练出更好的神经网络。

## 产品检测（predict.py）

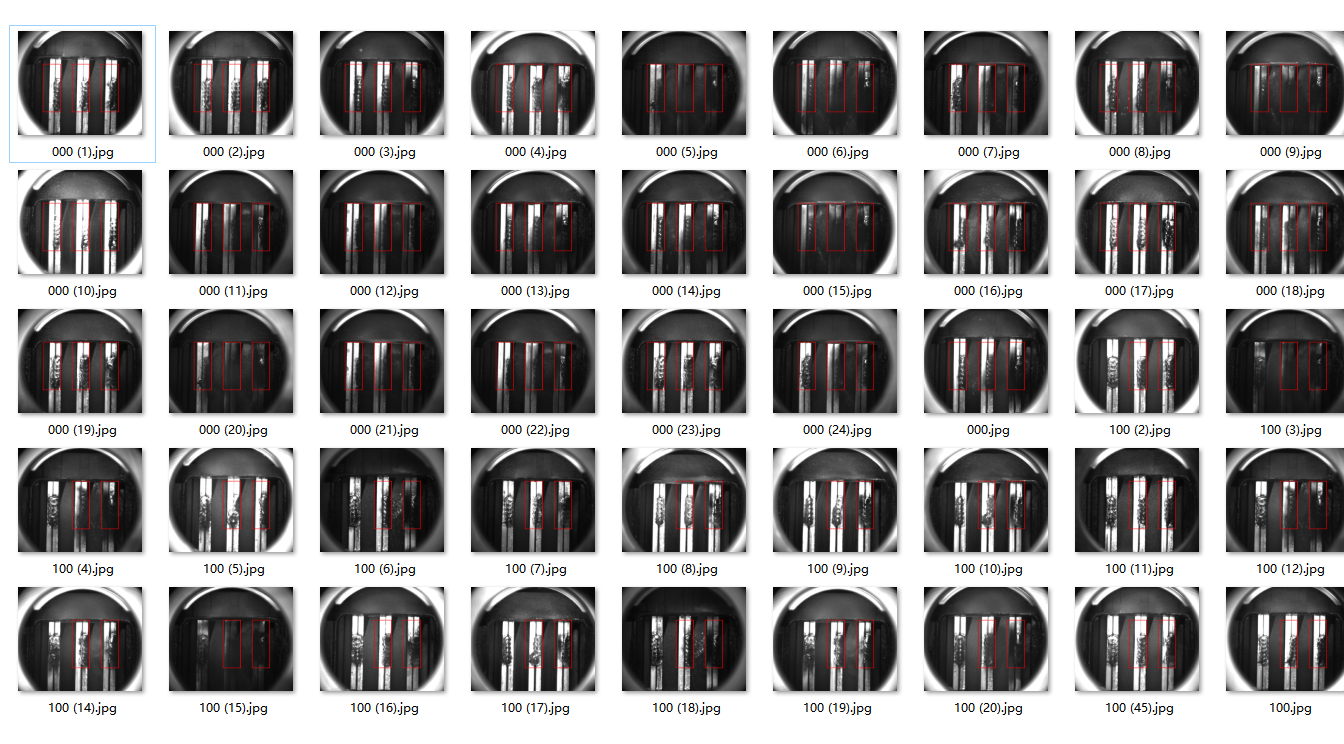
把产品图片放入check\_images 中（放入别的文件夹也可以，不过需要自行配置），然后运行程序，程序会加载之前训练好的神经网络，对预处理后的产品图片进行检测，最后把有缺陷的产品输出，并且用红框标记失败的焊接。

# 实验结果（result\_images ）

经过实验过程中的神经网络训练，最后得到训练集识别率达到100%，识别时间100张图片，程序开始运行🡺加载神经网络🡺图片预处理🡺缺陷检测🡺检测结果导出，总共花费16S时间，即每张图片的识别速度是0.16S。实验的代码我放在了github上（<https://github.com/guanyuqian/ai_homework>），检测的过程如下图所示：

## 图片结果输出

最后图片输出结果于“result\_images”内，把所有不合格的产品图片输出，并且圈出不合格的焊点，结果如下：



## 文本结果输出

最后文本结果输出于控制台内，把所有不合格的产品图片输出，并且指出出不合格的焊点位置，结果如下：

1. 开始时间 2018-10-08 13:55:07
2. Failed:000 (1).jpg left
3. Failed:000 (1).jpg middle
4. Failed:000 (1).jpg right
5. Failed:000 (10).jpg left
6. Failed:000 (10).jpg middle
7. Failed:000 (10).jpg right
8. Failed:000 (11).jpg left
9. Failed:000 (11).jpg middle
10. Failed:000 (11).jpg right
11. Failed:000 (12).jpg left
12. Failed:000 (12).jpg middle
13. Failed:000 (12).jpg right
14. Failed:000 (13).jpg left
15. Failed:000 (13).jpg middle
16. Failed:000 (13).jpg right
17. Failed:000 (14).jpg left
18. Failed:000 (14).jpg middle
19. Failed:000 (14).jpg right
20. Failed:000 (15).jpg left
21. Failed:000 (15).jpg middle
22. Failed:000 (15).jpg right
23. Failed:000 (16).jpg left
24. Failed:000 (16).jpg middle
25. Failed:000 (16).jpg right
26. Failed:000 (17).jpg left
27. Failed:000 (17).jpg middle
28. Failed:000 (17).jpg right
29. Failed:000 (18).jpg left
30. Failed:000 (18).jpg middle
31. Failed:000 (18).jpg right
32. Failed:000 (19).jpg left
33. Failed:000 (19).jpg middle
34. Failed:000 (19).jpg right
35. Failed:000 (2).jpg left
36. Failed:000 (2).jpg middle
37. Failed:000 (2).jpg right
38. Failed:000 (20).jpg left
39. Failed:000 (20).jpg middle
40. Failed:000 (20).jpg right
41. Failed:000 (21).jpg left
42. Failed:000 (21).jpg middle
43. Failed:000 (21).jpg right
44. Failed:000 (22).jpg left
45. Failed:000 (22).jpg middle
46. Failed:000 (22).jpg right
47. Failed:000 (23).jpg left
48. Failed:000 (23).jpg middle
49. Failed:000 (23).jpg right
50. Failed:000 (24).jpg left
51. Failed:000 (24).jpg middle
52. Failed:000 (24).jpg right
53. Failed:000 (3).jpg left
54. Failed:000 (3).jpg middle
55. Failed:000 (3).jpg right
56. Failed:000 (4).jpg left
57. Failed:000 (4).jpg middle
58. Failed:000 (4).jpg right
59. Failed:000 (5).jpg left
60. Failed:000 (5).jpg middle
61. Failed:000 (5).jpg right
62. Failed:000 (6).jpg left
63. Failed:000 (6).jpg middle
64. Failed:000 (6).jpg right
65. Failed:000 (7).jpg left
66. Failed:000 (7).jpg middle
67. Failed:000 (7).jpg right
68. Failed:000 (8).jpg left
69. Failed:000 (8).jpg middle
70. Failed:000 (8).jpg right
71. Failed:000 (9).jpg left
72. Failed:000 (9).jpg middle
73. Failed:000 (9).jpg right
74. Failed:000.jpg left
75. Failed:000.jpg middle
76. Failed:000.jpg right
77. Failed:100 (10).jpg middle
78. Failed:100 (10).jpg right
79. Failed:100 (11).jpg middle
80. Failed:100 (11).jpg right
81. Failed:100 (12).jpg middle
82. Failed:100 (12).jpg right
83. Failed:100 (14).jpg middle
84. Failed:100 (14).jpg right
85. Failed:100 (15).jpg middle
86. Failed:100 (15).jpg right
87. Failed:100 (16).jpg middle
88. Failed:100 (16).jpg right
89. Failed:100 (17).jpg middle
90. Failed:100 (17).jpg right
91. Failed:100 (18).jpg middle
92. Failed:100 (18).jpg right
93. Failed:100 (19).jpg middle
94. Failed:100 (19).jpg right
95. Failed:100 (2).jpg middle
96. Failed:100 (2).jpg right
97. Failed:100 (20).jpg middle
98. Failed:100 (20).jpg right
99. Failed:100 (3).jpg middle
100. Failed:100 (3).jpg right
101. Failed:100 (4).jpg middle
102. Failed:100 (4).jpg right
103. Failed:100 (45).jpg middle
104. Failed:100 (45).jpg right
105. Failed:100 (5).jpg middle
106. Failed:100 (5).jpg right
107. Failed:100 (6).jpg middle
108. Failed:100 (6).jpg right
109. Failed:100 (7).jpg middle
110. Failed:100 (7).jpg right
111. Failed:100 (8).jpg middle
112. Failed:100 (8).jpg right
113. Failed:100 (9).jpg middle
114. Failed:100 (9).jpg right
115. Failed:100.jpg middle
116. Failed:100.jpg right
117. 训练结束时间 2018-10-08 13:55:23

# 实验总结

## 关于方法选择

在老师布置下作业的时候，考虑实现方法，就有传统的机器学习和深度学习两条路径，由于老师给的数据集太少，曾一度想用传统方法实现。之后拜读了Andrew L. Beam的《[You can probably use deep learning even if your data isn't that big](http://beamandrew.github.io/deeplearning/2017/06/04/deep_learning_works.html)》，觉得用深度学习试一试，最后没想到得出的结果还不错，让我更能理解了利用神经网络的深度学习的强大之处。

## 数据集大小的思考

关于数据集我之前会因为老师给的数据集太少而相关数据集通过变换而丰富的方法，虽然最后因为没有变换的数据集得出的神经网络就达标，但是我相信在面对更多的产品检测的情况，丰富数据集的方式还是很有效果的，这个就等今后的工作中验证了。

## 深度学习入门的思考

搭建深度学习的过程中也学习了“[莫烦python](https://morvanzhou.github.io/)”的深度学习框架搭建，让我意识到了深度学习的入门门槛是很低的，也许这就是深度学习现在那么火的原因吧。

## 深度学习的基础知识掌握

之后在优化过程中的超参数的设置却让我十分为难，因为超参数的修改需要了解深度学习底层的基础知识，还有结合以往的深度学习经验。这样让我意识到深度学习这方向要想深入下去，还是得了解机器学习的基础知识，所以陈伟婷老师的“机器学习基础”还是很有必要学习的。