

# 实验 14- SearchByCNNFeatures

姓名：王煌基      学号：519030910100      班级：F1903004

## 一、实验概览

基于对于机器学习与深度学习的进一步认识，了解图像检索的基本方式以及以图搜图的过程，通过深度学习模型使得模型能够学习到图像的某种特定的特征来将图像转换成一个能够代表图像的向量，从而用向量之间的相似程度代表图像之间的相似程度，从而解决以图搜图的任务。理解模型训练过程的参量变化及参量意义，在此基础上，利用深度学习实现对初等函数的拟合以及对 CIFAR-10 图片进行分类。

## 二、实验环境

1. 个人笔记本电脑
2. 操作系统：windows10 专业版
3. 使用软件：Visual Studio Code; Docker Desktop

## 三、实验过程

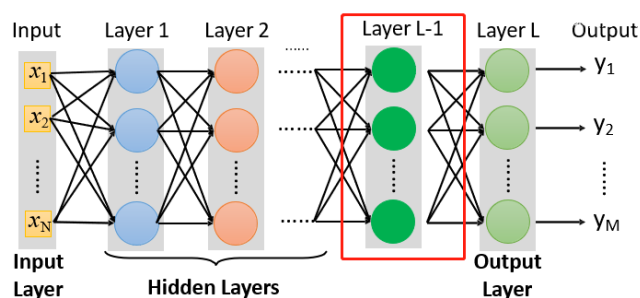
练习一：本次实验要求构建一个不小于 500 张的图像库，用深度学习模型提取图像的特征，并使用上一页 PPT 提到的方法，计算图片之间的相似度。

请使用一些不在图像库中的图片进行测试，完成以图搜图的检索任务。

报告中要求给出检索的结果，与被检索图片 Top5 相似的图片 and 排序，说明排序的方法，给出排序的得分情况。

### 【解】

本次实验其实与之前的图像处理部分的实验（如 LSH，如 SIFT）是有些相似的，具体表现在，二者都是通过某种方式，得到一定的特征之后，利用特征的比较（如欧氏距离，如向量夹角）来判断两张图是否相似的情况，因此，前面的实验其实可以为我们本次的实验提供一定的参考意义。本次实验的最大不同点在于本次实验利用到了深度学习模型进行特征提取（即提取特征的方法是比较独特的），简单的来说，可以用下图呈现：



即，本次实验是基于深度学习的过程实现的。现代神经网络的层数非常多（可能有 50 层甚至几百层），对这种多层神经网络的训练，可以认为就是深度学习。而其即将输出结果前的倒数第二层就可以作为输入的一个特征向量（由于经过了多层训练，可以默认正常情况下不会出现一样的图片），并且基于这个特征向量来实现具体的图片比较，在本次实验提供的参考程序（extract\_feature.py）中，通过把模型 model 的输出，可以验证我们的想法：

```
1. # 导入 ResNet50 模型，并加载预训练参数，定义处理图片的归一化操作和预处理方式。
2. print('Load model: ResNet50')
3. model = torch.hub.load('pytorch/vision', 'resnet50', pretrained=True)
4. # 这里的 model 是 resnet50 模型
5. print(model)
```

因为它的输出结果为：

```
1. ....
2. ResNet(
3.   (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
4.   (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
5.   (relu): ReLU(inplace=True)
6.   (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
7.   (layer1): Sequential(...)
8.   (layer2): Sequential(...)
9.   (layer3): Sequential(...)
10.  (layer4): Sequential(...)
11.  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
12.  (fc): Linear(in_features=2048, out_features=1000, bias=True)
13. )
14. '''
```

而程序的提取特征部分为：

```
1. def features(x):
2.     x = model.conv1(x)
3.     x = model.bn1(x)
4.     x = model.relu(x)
5.     x = model.maxpool(x)
6.     x = model.layer1(x)
7.     x = model.layer2(x)
8.     x = model.layer3(x)
9.     x = model.layer4(x)
10.    x = model.avgpool(x)
11.
12.    return x
```

缺少了最后一层的 model.fc(x)，说明我们的想法是正确的。

因此，通过对程序这样的分析之后，易知，我们现在需要实现的流程，应该是：

1. 读入需检测图片，计算其特征向量

2. 读入待匹配数据集，计算各自的特征向量

3. 取二者欧氏距离（角度），取前五最小匹配结果，即所求结果。

通过这样的流程，我们可以很好地进行本次实验。

### 1. 读入需检测图片，计算其特征向量

这里我选择的图片是 103.jpg（如右图所示）进行测试。



103.jpg

首先对第一张图片进行输入检测，利用类似于 panda.jpg 的操作进行预处理：

```
1. # 读入图片，并使用之前定义好的处理方式(trans)处理图片，为送入模型提特征做准备。
2. print('Prepare image data!')
3. test_image = default_loader('103.jpg')
4. # 获取 PIL 图
5. input_image = trans(test_image)
6. input_image = torch.unsqueeze(input_image, 0)
7.
8.
9. print('Extract features!')
10. start = time.time()
11. # 提取图片的 feature 特征
12. image_feature = features(input_image)
13. # 并且利用 numpy 模式存储
14. image_feature = image_feature.detach().numpy()
15. print('Time for extracting features: {:.2f}'.format(time.time() - start))
16.
17. # image_feature 是待测图片的特征向量
18. print('Save features!')
19. # 存储特征
20. np.save('features.npy', image_feature)
```

### 2. 读入待匹配数据集，计算各自的特征向量

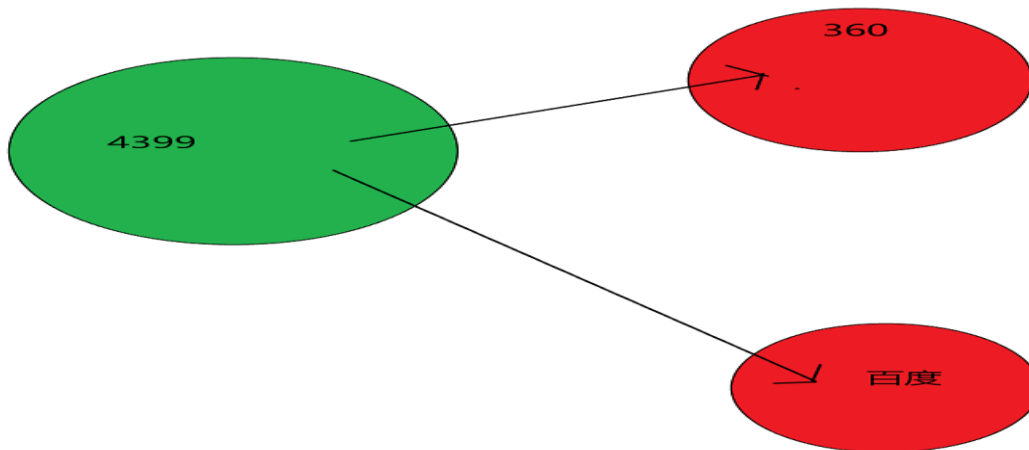
这里的一大难点在于数据图的获取，本质上是对前面实验内容的整合。

首先，选定特定的网站进行爬取。这里我本来选择的是 17yy 小游戏网站进行爬取，但由于爬取次数过多，已经短时间内被禁止再次访问，故为了继续进行实验，我选择 4399 小游戏的网站继续爬取。

基于之前的实验，我们在爬取每一个网页的时候，通过对网页中所有的 img 类下的 src 标签即链接进行获取，并且注意到 alt 标签就是图片的相关标题（如果 alt 存在），因此可以很容易得到图片与图片名称的具体关系。

而网页的链接的获取则是通过 a 标签的“外链” (outlinks) 进行，并且由于在具体实验的时候容易爬取到很多无关网站，例如，我们的目的是在 4399 的网站下爬取各种小游

戏的图片，然而由于其网站本身就存在如赞助、广告等外链，若爬取则对我们是非常不利的，可以用下图解释：



我们的原意是在绿色的 4399 网站闭环内查找所有相关的到 4399 内部网站的链接，但是由于外链的存在，我们如果没有加以处理，则容易出现爬取时在如 360、百度等广告内进行内部循环，这是非常低效的，故可以利用域名限定的方法进行爬取以便提高效率。

```
1. new_site = parse.urlparse(t).hostname
2. # 必须确保准备爬取的网页链接与初始链接在同一域名
3. if new_site == basic_site:
4.     if t not in links:
5.         links.append(t)
```

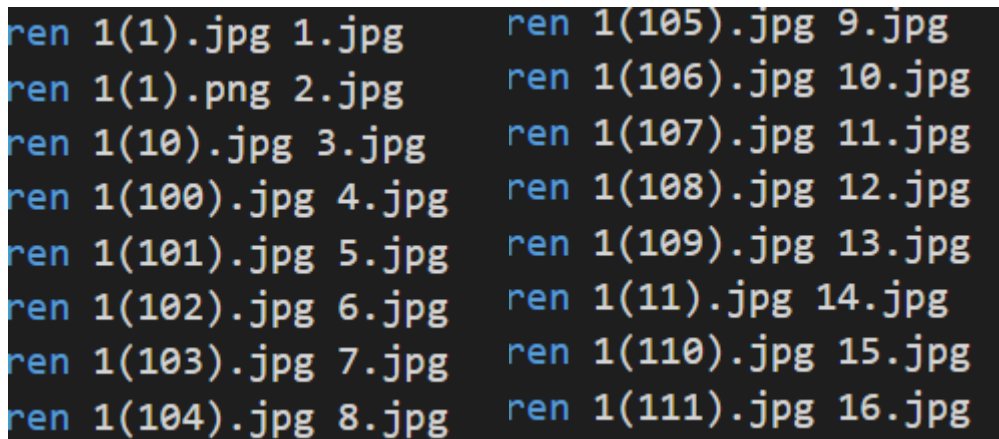
爬取完所有图片链接之后，通过对其后缀不同的图片类型进行判定来运算，将图片分类为 jpg 类和 png 类，防止在图片读取时存在不可知因素的问题：

```
1. def download_photo(page,cnt):
2.     .....
3.     # 判定该图片类型是 png 还是 jpg
4.     if '.png' in page:
5.         print("png")
6.         filename = get_name(page) + str(cnt) + '.png' # 将网址变成合法的文件名
7.     else:
8.         print("jpg")
9.         filename = get_name(page) + str(cnt) + '.jpg' # 将网址变成合法的文件名
10.    if not os.path.exists(folder): # 如果文件夹不存在则新建
11.        os.mkdir(folder)
12.    # 由于是媒体文件，需要利用 wb 格式存储
13.    f = open(os.path.join(folder, filename), 'wb')
14.    f.write(content) # 将网页存入文件
15.    #f.write(content) # 将网页存入文件
16.    f.close()
```

最后可以在本地端得到大量的图片集合。

然而，大量的图片中存在一些无法读取或者无法显示的图片，由于很多的网站本身年代久远，引用的图片可能或多或少存在丢失的情况，故这样无法显示的现象是正常合理的。我们只需要处理正常可用的图片即可。

由于图片名字参差不齐，可以通过批处理文件或者通过 python 的 os 库函数、Path 库函数等方式进行读取来方便操作，这里我通过批处理文件方式，先确保文件名统一为 i.jpg（i 为编号）从而方便进行操作（下图为 bat 文件的具体实现，作用即重命名文



件）。

然后便可以将编号为 1 到 569 的 jpg 图片读入进行操作了（本次选取了 569 张图片，实际上可以更多，仅取决于我们的限制）。

```
1. for i in range(1,570):
2.     filename = str(i) + '.jpg'
3.     print('Prepare image data: ' + filename + " !")
4.     temp_image = default_loader(os.path.join('testpic',filename))
5.     test_image = trans(temp_image)
6.     test_image = torch.unsqueeze(test_image, 0)
7.     # print('Extract features!')
8.     test_feature = features(test_image)
9.     test_feature = test_feature.detach().numpy()
```

这一块特征提取实际上是与前面的待匹配图片一样的，故不多做注释说明。

### 3. 取二者欧氏距离（角度），取前五最小匹配结果，即所求结果。

这里首先需要实现两个特征的欧氏距离或者欧式角度的函数：

由于特征的保存方式的一个带有多重嵌套的 numpy 矩阵，在计算时不能直接利用语句 for i in feature: 去操作，通过反复调试，发现可以利用这样的方式进行操作：

求欧氏距离：

```
1. def dist(feature1,feature2):
2.     # 需要归一化，否则容易过大
3.     dis, t1, t2 = 0, 0, 0
4.     for i in feature1[0]:
```

```

5.         t1 += i[0][0]**2
6.     t1 = t1**0.5
7.     feature11 = [i[0][0]*1.0/t1 for i in feature1[0]]
8.     for i in feature2[0]:
9.         t2 += i[0][0]**2
10.    t2 = t2**0.5
11.    feature22 = [i[0][0]*1.0/t2 for i in feature2[0]]
12.    for i in range(len(feature11)):
13.        dis += (feature11[i]-feature22[i])**2
14.    dis = dis**0.5
15.    return dis

```

求角度：

```

1. def angle(feature1,feature2):
2.     # cosθ=
3.     #  $x \cdot y = |x| |y| \cos\theta$ 
4.     cnt, t1, t2 = 0, 0, 0
5.     for i in feature1[0]:
6.         t1 += i[0][0]**2
7.     t1 = t1**0.5
8.     feature11 = [i[0][0] for i in feature1[0]]
9.     for i in feature2[0]:
10.        t2 += i[0][0]**2
11.    t2 = t2**0.5
12.    feature22 = [i[0][0] for i in feature2[0]]
13.    for i in range(len(feature22)):
14.        cnt += feature11[i]*feature22[i]
15.    cnt /= t1
16.    cnt /= t2
17.    return cnt

```

对于这一些衡量“匹配程度”的特殊值进行计算后，将其保存在列表中，并且进行冒泡排序来得到最小值，从而得到最佳匹配度的前五名。

这里利用的是最基础的冒泡排序（由于数据规模小，可以很方便使用，若数据规模较大，则可以使用如优先队列在内的排序方式进行优化到  $O(n\log n)$  水平），且若需要调用的函数为欧氏距离，则符号为<，若为角度，则符号为>（因为角度为越贴近 1 越好）。

```

1. distance.append([dist(image_feature,test_feature),filename])
2.
3. for i in range(len(distance)):
4.     for j in range(len(distance)):
5.         if(distance[i][0]<distance[j][0]):
6.             distance[i], distance[j] = distance[j], distance[i]
7. for i in range(5):
8.     print(distance[i])

```

那么我们的总体实验就完成了。



## 4. 实验结果

当利用欧氏距离进行匹配时可以得到下面的结果：

```
Time for extracting features: 127.29  
[0.0, '103.jpg']  
[0.08506033507294919, '111.jpg']  
[0.08570788733817539, '202.jpg']  
[0.0857562944532484, '234.jpg']  
[0.08595153197262312, '104.jpg']
```

其所代表的图片为（从左往右依次）：



当利用角度计算进行匹配时可以得到下面的结果：

```
Time for extracting features: 121.36  
[1.0000000005354401, '103.jpg']  
[0.9963823695575088, '111.jpg']  
[0.9963270793523509, '202.jpg']  
[0.9963229289768338, '234.jpg']  
[0.9963061670931651, '104.jpg']
```

其所代表的图片为（从左往右依次）：



因此，可以认为，本次实验完成得比较好。

## 四、实验问题及思考

### 1. 大量的图片无法读入的现象

在具体上机实验的时候，我遇到了一个问题，即，存在大量的图片无法读入的现象，即目录下确实存在相应的图片，但是在文件读入(default\_loader)的时候却始终无法读取（读入为空），向助教请教之后最初以为是 jpg、png 的问题，但经过调试发现并非全部因为该种情况（该种情况的原因在于图片不完全是 .jpg 格式，可能只是后缀是 .jpg 而我实验中在爬取图片时将所有图片强行转换成了 jpg 格式的图片。这里的解决方案是利用到要用到 PIL 中的 `Image.open("filepath").convert("RGB")`。），仍旧无法解决问题。

后来通过对成功图像与失败图像的对比中我发现，可能图片大小的问题。因为我注意到这一类不能识别的图片都有一个显著的特点，即图片过小，基本都在 1KB~5KB 之间，而能够成功进行实验的大多在 6KB 以上，因此我对 default\_loader 函数所能处理的图像大小产生了一定的思考。通过查阅相关的资料文档以及询问助教，我了解到，在深度学习中，正常 resnet50 网络提特征的输入图片大小是 224\*224 的，本身而言图片太小了会导致特征可能不容易提取甚至直接无法提取，进而无法生成 PIL 格式的图片而出现了查找不到文件的错误。下图都是图片大小小于 5KB 的图片（接近于 2KB~3KB）



2. 我们在 lab12 中学习了 LSH 加速检索，能否使用本次实验提取的 CNN 特征进行 LSH 检索？

根据 LSH 的原理，我们本质上是可以利用 LSH 进行 CNN 特征提取的，理想操作如下：

CNN 提取特征→特征向量降维→LSH 检索→返回搜索结果

LSH 检索的具体实现应该为：

创建哈希表→HASH 索引→汉明距离判定相似性

需要实现降维的原因是本次实验中获得的特征向量维度是 2048 维，并且在归一化处理的情况下，如果仍旧按照  $0 \sim 0.33$ ,  $0.33 \sim 0.66$ ,  $0.66 \sim 1$  来划分是相当危险的，故需要进行降维处理，只是处理起来较为复杂，但是仍旧可以实现。

由于时间紧迫，来不及对这一块进行具体的实现完成，只是查阅相关资料，发现了我这个想法确实是一个可行的思路，相关资料为：《[深度学习 - 实战项目] 以图搜图 Resnet+LSH-特征编码/图像检索/相似度计算》，链接为：

[https://blog.csdn.net/weixin\\_41809530/article/details/109258984](https://blog.csdn.net/weixin_41809530/article/details/109258984)。

## 五、实验体会

这是本课程的最后一次实验，将近学期末，我的收获非常丰富。

在本次实验中，深度学习可以被用来作为一个图像处理特征向量提取的工具，是一种全新的想法，让我对深度学习有了较为浅薄的认知。虽然本次实验中遇到了比较棘手的问题，但是在多方向的思考、求助之后问题终于得到了解决，内心十分具有成就感。

希望在大作业中，我能和我们小组一齐协力交出一份最好的答卷！