|  |  |  |
| --- | --- | --- |
| **计算机视觉实验报告** | | |
| 姓名： 吴清杰 | 学号：ZF2021347 | 班级： ZF20213 |
| **试验目的：**  掌握图像检索系统的实现.  利用现有的图像库中检索出最相似的图片 | | |
| **试验内容：**   * + - 提取图库中所有图片特征。     - 提取查询图片特征。     - 查询图片特征与图库图片特征对比,输出打分最高的前三的图片。 | | |
| **试验步骤：**  **(要求：说明解决思路，执行结果页面进行截图，对实验进行总结和分析；还有需要报告作者姓名学号信息的登录图，如下图所示)**   1. **解决思路**   采用VGG16模型,将图库中的图片所有特征抽取出来并存储起来,同时拿到待检索的图片的向量,将图库的向量矩阵和待检索向量相乘,获取结果之后,进行评分排序,并输出   1. **完成代码**   Bulid\_index.py  curPath = os.path.abspath(os.path.dirname(\_\_file\_\_)) rootPath = curPath[:curPath.find("2021CVORDL\\") + len("2021CVORDL\\")] # dataPath = rootPath + "Input/MLWorkHome/experiment3/data.txt"  ap = argparse.ArgumentParser() ap.add\_argument("-database", required=True,  help="Path to database which contains images to be indexed") ap.add\_argument("-index", required=True,  help="Name of index file") args = vars(ap.parse\_args()) args["database"] = rootPath + args["database"] args["index"] = rootPath + args["index"]  '''  Returns a list of filenames for all jpg images in a directory.  '''   def get\_imlist(path):  fs = []  for root, dirs, files in os.walk(path):  root  for f in files:  if (f.endswith(".jpg") or f.endswith(".png")):  yield os.path.join(root, f)  # fs.append(os.path.join(root, f))   '''  Extract features and index the images  -database src\main\resources\com\wqj\cv\bighomework\database -index src/featureCNN.h5 ''' if \_\_name\_\_ == "\_\_main\_\_":   db = args["database"]  img\_list = list(get\_imlist(db))  # img\_list = get\_imlist("D:\develop\_disk\python/2021CVORDL\src\main\com\wqj\cv")  for img in img\_list:  print(img)   print("--------------------------------------------------")  print(" feature extraction starts")  print("--------------------------------------------------")   feats = []  names = []   model = VGGNet()  for i, img\_path in enumerate(img\_list):  norm\_feat = model.extract\_feat(img\_path)  img\_name = os.path.split(img\_path)[1]  feats.append(norm\_feat)  names.append(img\_name.encode())  print("extracting feature from image No. %d , %d images in total" % ((i + 1), len(img\_list)))   feats = np.array(feats)  # directory for storing extracted features  output = args["index"]   print("--------------------------------------------------")  print(" writing feature extraction results ...")  print("--------------------------------------------------")   h5f = h5py.File(output, 'w')  h5f.create\_dataset('dataset\_1', data=feats)  h5f.create\_dataset('dataset\_2', data=names)  h5f.close()  VGG.py  class VGGNet:  def \_\_init\_\_(self):  # weights: 'imagenet'  # pooling: 'max' or 'avg'  # input\_shape: (width, height, 3), width and height should >= 48  self.input\_shape = (224, 224, 3)  self.weight = 'imagenet'  self.pooling = 'max'  self.model = VGG16(weights = self.weight, input\_shape = (self.input\_shape[0], self.input\_shape[1], self.input\_shape[2]), pooling = self.pooling, include\_top = False)  self.model.predict(np.zeros((1, 224, 224 , 3)))   '''  Use vgg16 model to extract features  Output normalized feature vector  '''  def extract\_feat(self, img\_path):  img = image.load\_img(img\_path, target\_size=(self.input\_shape[0], self.input\_shape[1]))  img = image.img\_to\_array(img)  img = np.expand\_dims(img, axis=0)  img = preprocess\_input(img)  feat = self.model.predict(img)  norm\_feat = feat[0]/LA.norm(feat[0])  return norm\_feat  query.py  curPath = os.path.abspath(os.path.dirname(\_\_file\_\_)) rootPath = curPath[:curPath.find("2021CVORDL\\") + len("2021CVORDL\\")] # dataPath = rootPath + "Input/MLWorkHome/experiment3/data.txt" """ -query src\main\resources\com\wqj\cv\bighomework\database\674.jpg -index src/featureCNN.h5 -result src\main\resources\com\wqj\cv\bighomework\database """  ap = argparse.ArgumentParser() ap.add\_argument("-query", required=True,  help="Path to query which contains image to be queried") ap.add\_argument("-index", required=True,  help="Path to index") ap.add\_argument("-result", required=True,  help="Path for output retrieved images") args = vars(ap.parse\_args()) args["index"] = rootPath + args["index"] args["query"] = rootPath + args["query"] args["result"] = rootPath + args["result"]  # read in indexed images' feature vectors and corresponding image names h5f = h5py.File(args["index"], 'r') feats = h5f['dataset\_1'][:] imgNames = h5f['dataset\_2'][:] h5f.close()  print("--------------------------------------------------") print(" searching starts") print("--------------------------------------------------")  # read and show query image queryDir = args["query"] if (str(queryDir).endswith("png")):   queryImg = mpimg.imread(queryDir,0) else:  queryImg = mpimg.imread(queryDir) plt.figure(0) plt.title("Query Image") plt.imshow(queryImg) plt.show()  # init VGGNet16 model model = VGGNet()  # extract query image's feature, compute simlarity score and sort queryVec = model.extract\_feat(queryDir) scores = np.dot(queryVec, feats.T) rank\_ID = np.argsort(scores)[::-1] rank\_score = scores[rank\_ID] # print rank\_ID # print rank\_score   # number of top retrieved images to show maxres = 3 imlist = [imgNames[index] for i, index in enumerate(rank\_ID[0:maxres])] print("top %d images in order are: " % maxres, imlist)  # show top #maxres retrieved result one by one for i, im in enumerate(imlist):  if (str(queryDir).endswith("png")):  image = mpimg.imread(args["result"] + "/" + str(im, encoding='utf-8'),0)  else:  image = mpimg.imread(args["result"] + "/" + str(im, encoding='utf-8'))  plt.figure(i + 1)  plt.title("search output %d" % (i + 1))  plt.imshow(image)  plt.show()   1. 执行结果截图   待查询图片    查询出第一相似图片是本身:    剩下的分别为第二第三:    **4. 实验总结分析**  先抽取图库中所有图片的特征,然后将图片特征进行相似度相乘对比,取相似度较高的图片,从该实验看来,采用抽取特诊再相似度对比的比较符合实际 | | |
| **三、对于实验的改进意见（选填）**  当图片数据集比较大的时候,采用向量相乘的办法,对显存消耗太大,计算机较慢,可设置倒排索引的方式,将图片的特征归类,采用倒索引方式,聚合所有特征,同时记录每个特征所关联的图片,在通过累加相同的特征值个数的方式,进行打分,取出分值较高的图片编号即可. | | |