Introduction to CV Projects

- Preparation
- Image Filtering and Hybrid Images
- Local Feature Match
- Scene Recognition with Bag of Words
- Face Detection with a Sliding Window
- Fish Detection with Deep Learning

Project 0: Prepare for requirements

配置环境：推荐使用VS Code + Anaconda

- Python3.x
- Jupyter-notebook
- Pytorch >= 1.2
- scikit-image
- scikit-learn
- ……

Project 1: Image Filtering and Hybrid Images

目的：

对不同图像分别进行高通和低通滤波，再混合图片

操作关键点：

1. 编写my_filter()函数，并利用滤波器对图片进行滤波
2. 在part1部分对图像进行高低通滤波
3. 在part2部分进行图像的融合

代码： proj1/code/student.py

1. my_imfilter()

   利用滤波函数操作图像

2. gen_hybrid_image()

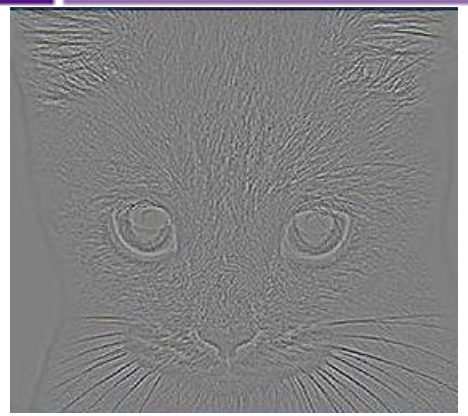
   生成高低通分量滤除图片

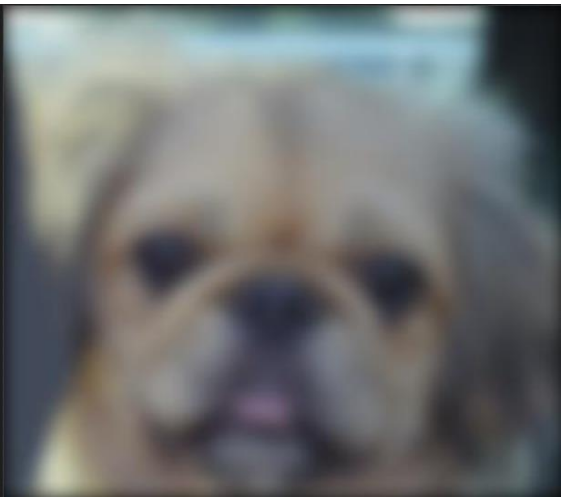
   图像融合

Project 1: Image Filtering and Hybrid Images


```python
def my_imfilter(image, filter):
  """
  Your function should meet the requirements laid out on the project webpage.
  Apply a filter to an image. Return the filtered image.
  Inputs:
  - image -> numpy nd-array of dim (m, n, c)
  - filter -> numpy nd-array of odd dim (k, l)
  Returns
  - filtered_image -> numpy nd-array of dim (m, n, c)
  Errors if:
  - filter has any even dimension -> raise an Exception with a suitable error message.
  """
  filtered_image = np.asarray([0])

  ##################
  # Your code here #
  raise NotImplementedError('my_imfilter function in student.py needs to be implemented')
  ##################

  return filtered_image
```

Project 1: Image Filtering and Hybrid Images





Project 2: Local Feature Match

目的：

SIFT特征的具体细节理解和简单的距离计算

操作关键点：

1. Harris兴趣点提取
   1.1 实现Harris角点检测算法（NMS）
   1.2 输入image（不一定是灰度图），返回一组角点坐标
2. 特征描述
   2.1 实现SIFT算法的特征描述部分（高斯模糊去噪、Norm）
   2.2 使用角点作为SIFT的关键点
   3.3 （optional：你也可以使用DOG）
3. 特征匹配
   4.1 欧式距离
   4.2 任意两点进行特征间的距离计算，
   4.3 不正确匹配点去除：阈值、最近距离与次近距离的比值。

代码位置：

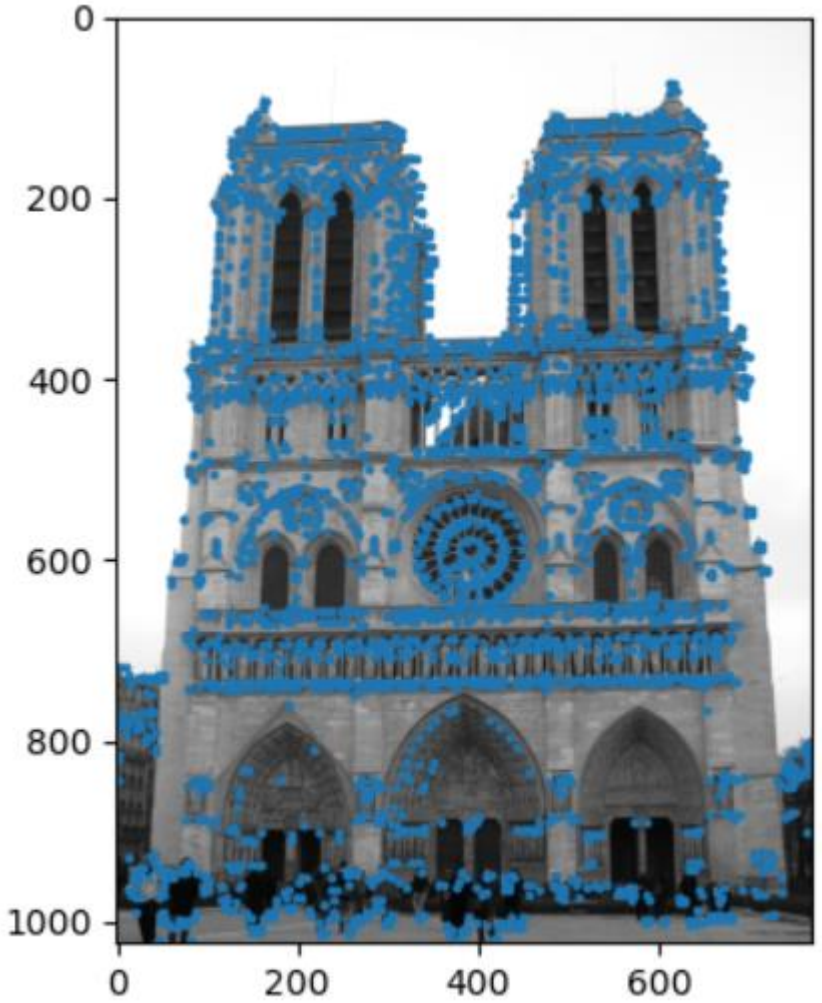
1. DoG关键点提取或者Harris关键点提取
   Proj2/code/student.py /get_interest_points()

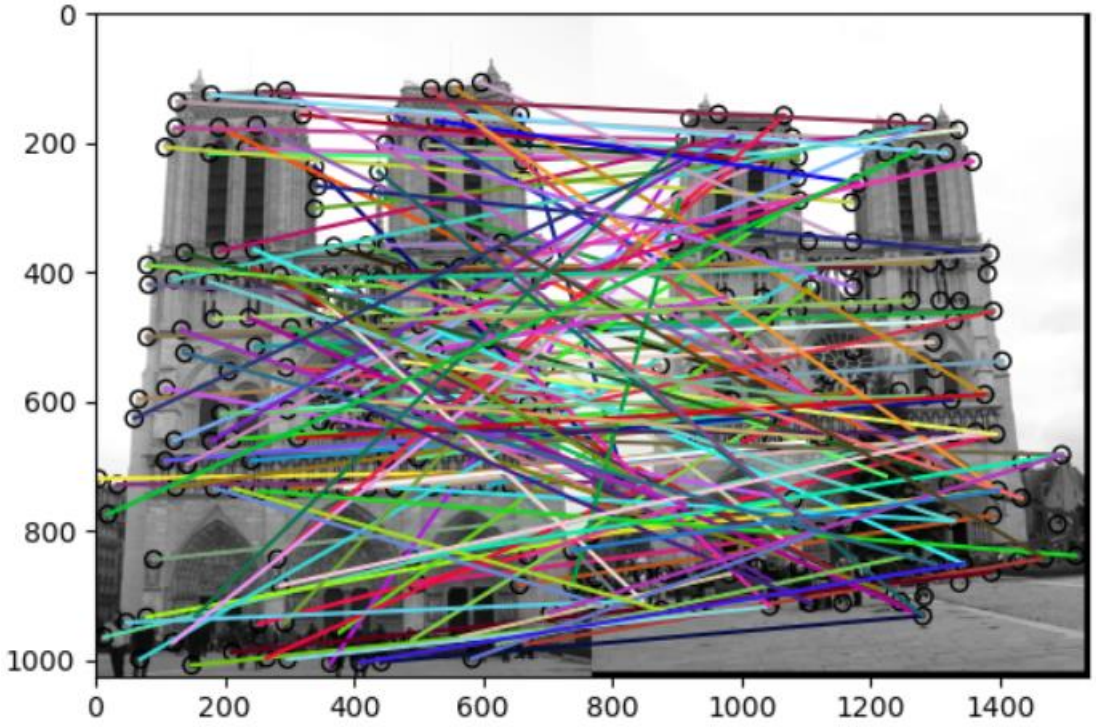
2. 关键点方向确定。
   Proj2/code/student.py / get_features()

3. 特征描述
   Proj2/code/student.py / get_features()

4. 特征匹配
   Proj2/code/student.py / match_features()

Project 2: Local Feature Match

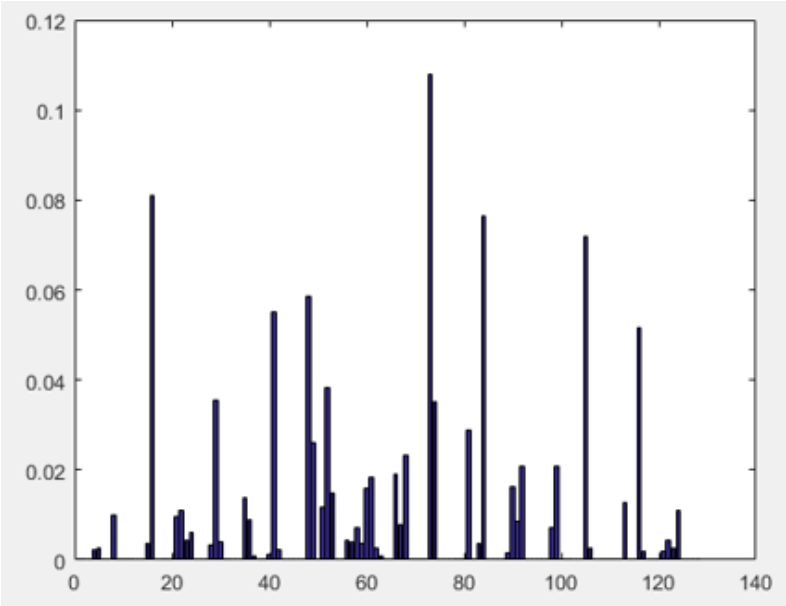




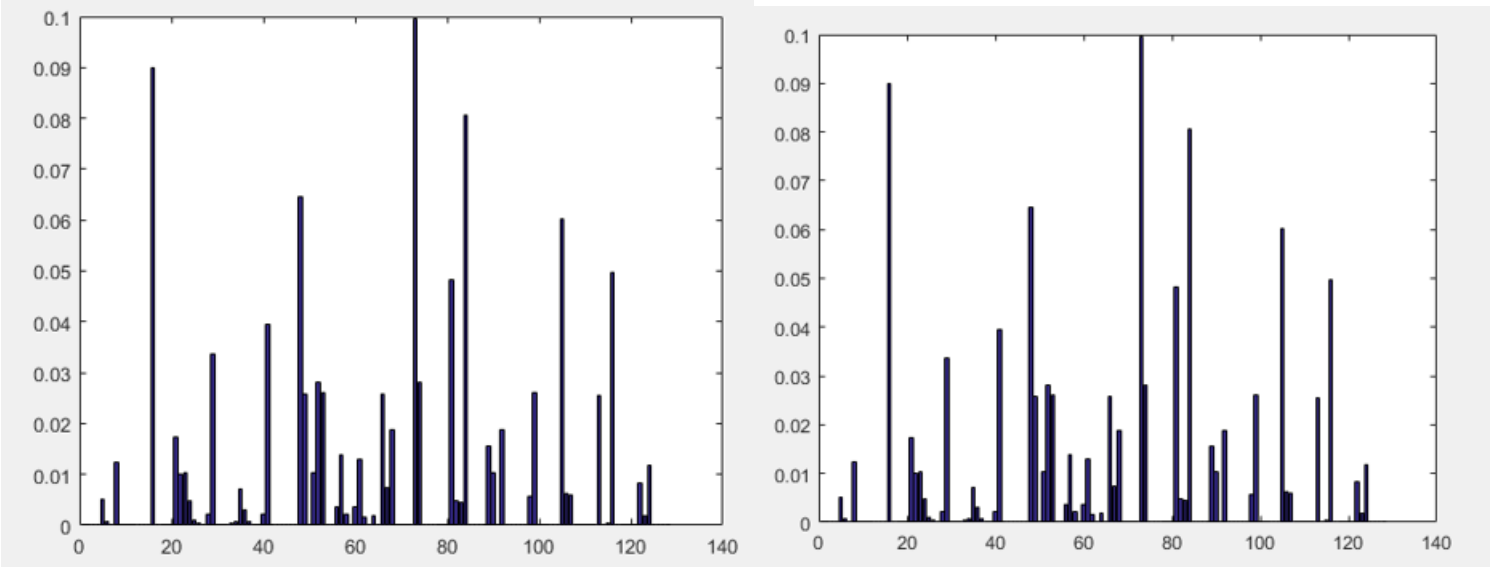

```
Matches: 155
91.0 total good matches, 58 total bad matches.
61.07382550335571% precision
64% accuracy (top 100)
Vizualizing...
```

Project 2: Local Feature Match

第一张图某点



第二张图前两个匹配点



Project 3: Scene recognition with bag of words

目的：

场景识别。特征提取+分类器构建和使用

操作关键点：

1. 图像词袋特征表示
   1.1 tiny特征
   1.2 BOW特征（SIFT or HoG）
   词袋构建：将从训练集获取的特征进行k-means聚类。K的取值决定了后续的效果

2. 分类器构建
   2.1 KNN classifier
   2.2 SVM

代码：proj3/code/student.py

1. get_tiny_images()
2. Build_vocabulary()
   包括图片的sift特征提取和词袋构建
3. get_bags_of_words()
   根据词袋获取测试图片的直方图特征表示
4. SVM_classify()

5. Nearest_neighbor_classify()

Project 3: Scene recognition with bag of words

具体操作步骤

1. Tiny + Nearest Neighbor

2. Bags of SIFT + Nearest Neighbor

3. Bags of SIFT + SVM

3.1 Bags of SIFT

3.1.1 提取训练集中每个图片的SIFT特征—可能不同的图片提取出来的sift特征个数不同

3.1.2 将训练集中的所有图片的SIFT特征进行KMeans聚类，每个类即可以认为是一个bag。假设K=500

3.1.3 统计训练集每张图片的SIFT特征落在不同bag的次数——每张图片的特征表示为500x1

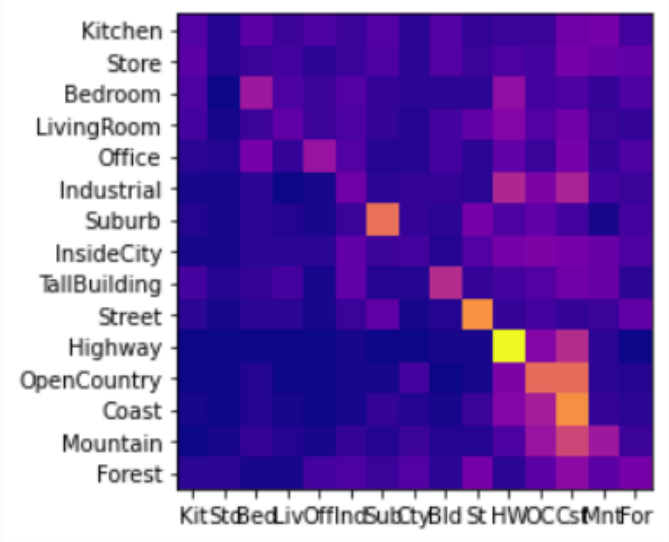
——利用Nearest 来判断某个SIFT特征的归类

3.2 SVM训练

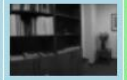
推荐使用sklearn.svm.LinearSVC

Tiny + KNN

scene classification results visualization

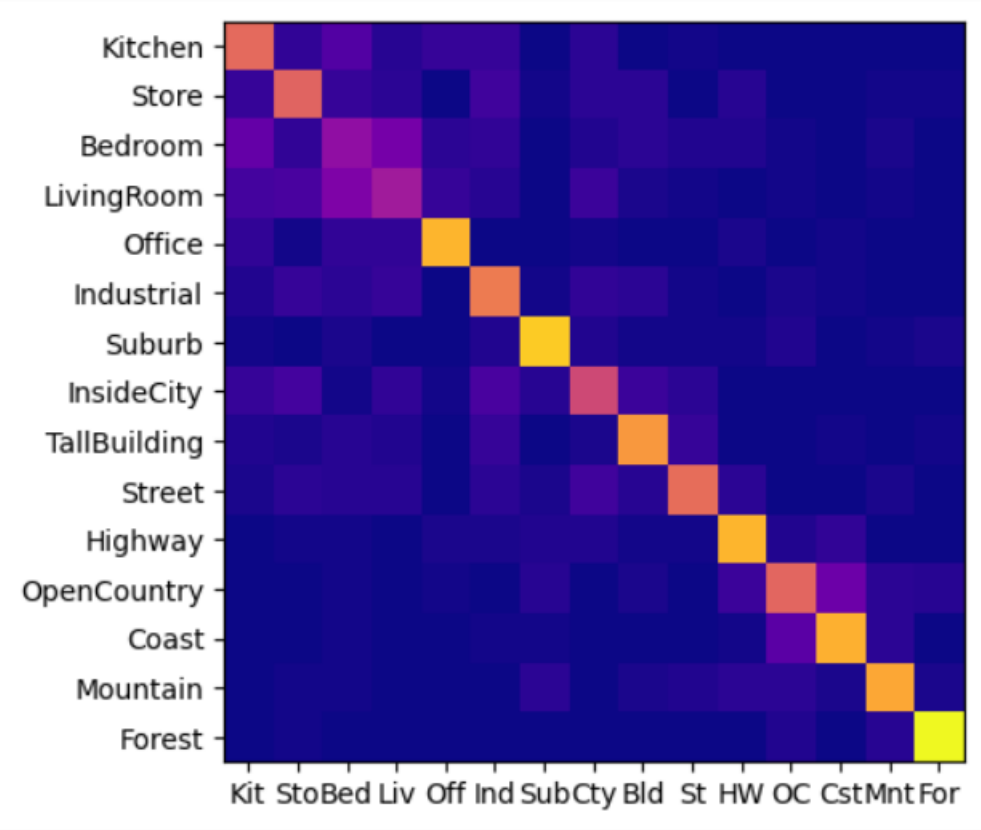


Accuracy (mean of diagonal of confusion matrix) is 0.227

Category name	Accuracy	Sample training images		Sample true positives		False positives with true label		False negatives with wrong predicted label	
Kitchen	0.080					 TallBuilding	 TallBuilding	 TallBuilding	 Mountain
Store	0.020					 Kitchen	 TallBuilding	 LivingRoom	 OpenCountry
Bedroom	0.190					 Mountain	 InsideCity	 Industrial	 Office
LivingRoom	0.100					 Bedroom	 Store	 Coast	 TallBuilding
Office	0.180					 Kitchen	 Kitchen	 Forest	OpenCountry

BOW + SVM

scene classification results visualization



Accuracy (mean of diagonal of confusion matrix) is 0.629

Project 4: Face Detection with a sliding window

目的：

基于HOG和多尺度滑动窗口实现人脸检测算法。

操作关键点：

1. 分类器训练
   - 1.1 正负样本的HoG特征获取
     - 正样本：人脸图片
     - 负样本：不包含脸的背景图
   - 1.2 图片HoG特征计算

2. 多尺度窗口滑动
   - 2.1 从图像中滑动取出预选框
   - 2.2 可能需要对图像进行缩放，简历尺度金字塔

1. 提取正负样本
2. 训练
3. 评估训练成果
4. 挖掘难样本
5. 在测试集上运行并可视化结果

Project 4: Face Detection with a sliding window

具体操作步骤

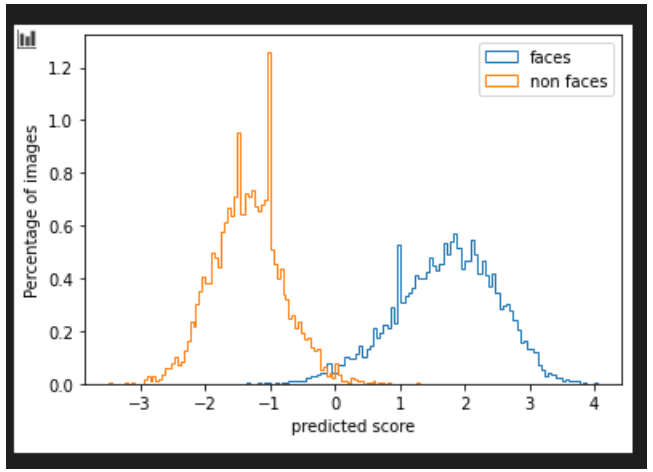
1. 提取训练图片的HoG特征。
2. 利用训练图片的HoG特征训练分类器。比如前面提到的SVM分类器
3. 计算测试图片的HoG特征图。按照step滑动窗口，利用分类器筛选出人脸窗口
4. 对HoG特征图降采样，按照step滑动窗口，筛选出人脸窗口，并按照降采样的尺度还原到原图。
5. 利用NMS移除重叠候选框
6. 下一张图

代码 proj4/code/student.py

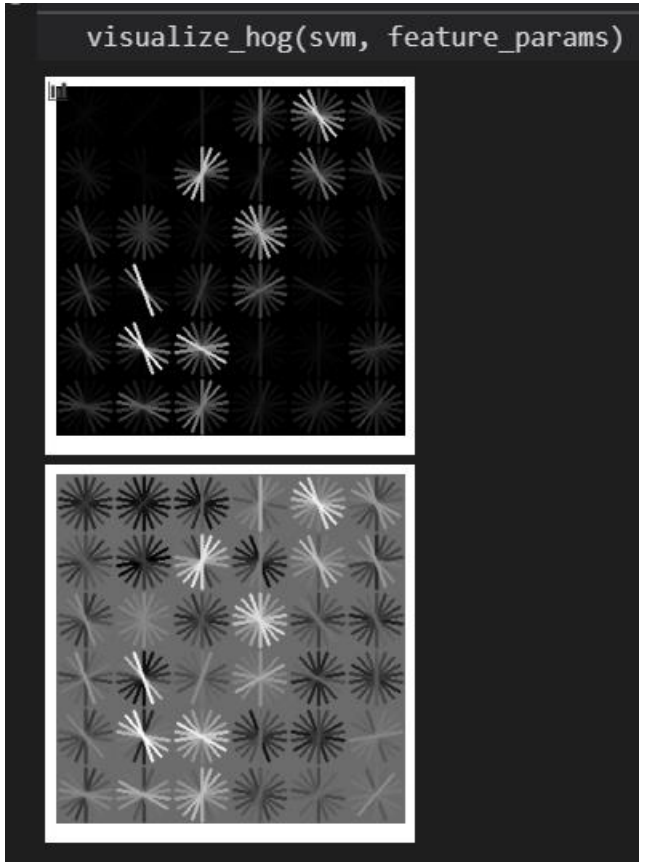
1. get_positive_features()　加载正类样本和计算这些样本的特征 。正负类样本路径在文件中都有
2. get_random_negative_features() 随即加载负样本并计算这些样本特征
3. train_classifier() 利用SVM训练线型分类器
4. mine_hard_negs() 提取false-positive prediction.特征
5. run_detector() 测试图多尺度寻找人脸，并且利用NMS移除部分候选框。

Project 4: Face Detection with a sliding window

查看分类器训练结果

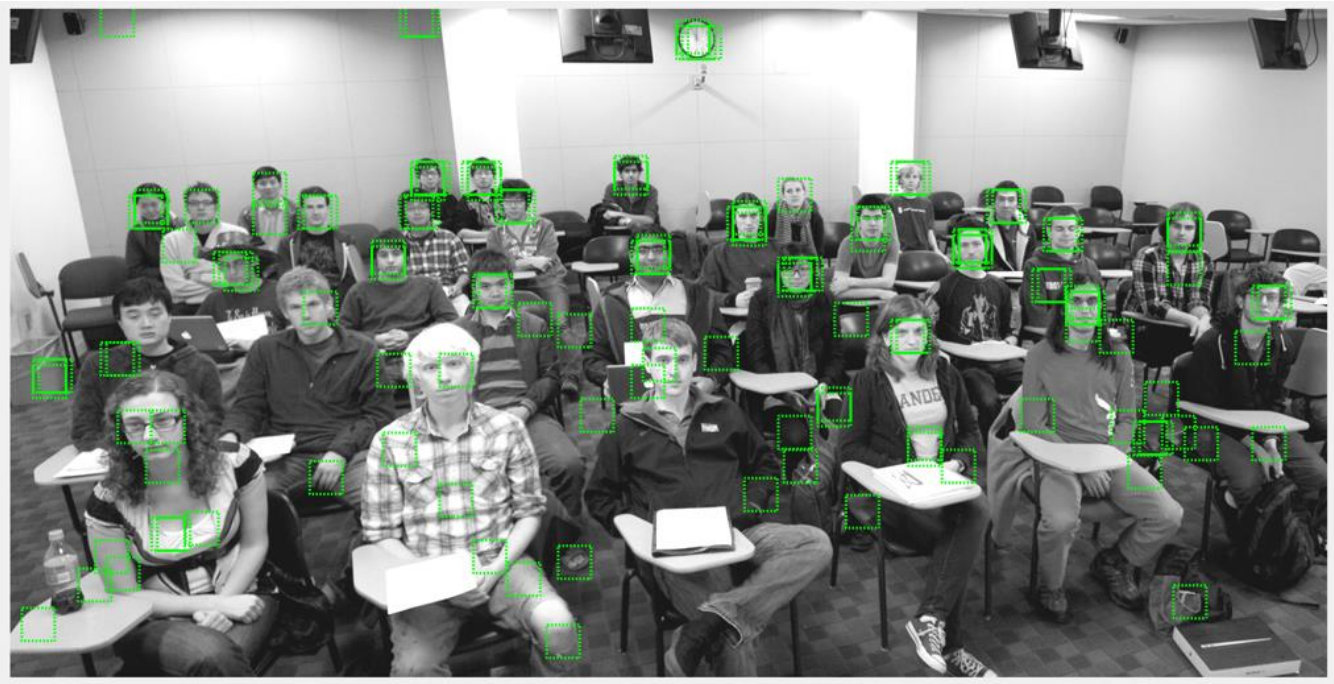



```
Accuracy = 98.267%
True Positive rate = 97.795%
False Positive rate = 1.100%
True Negative rate = 98.900%
False Negative rate = 2.205%
```

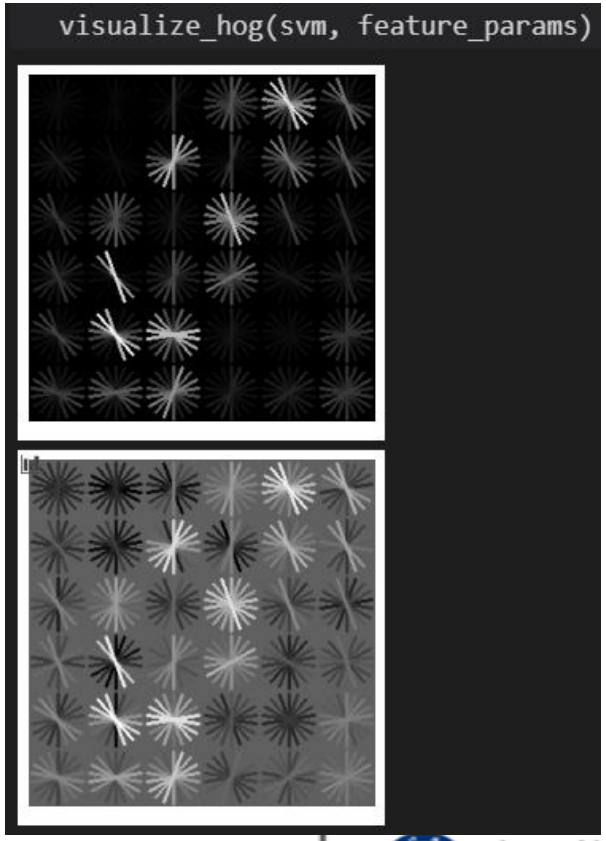


Project 4: Face Detection with a sliding window

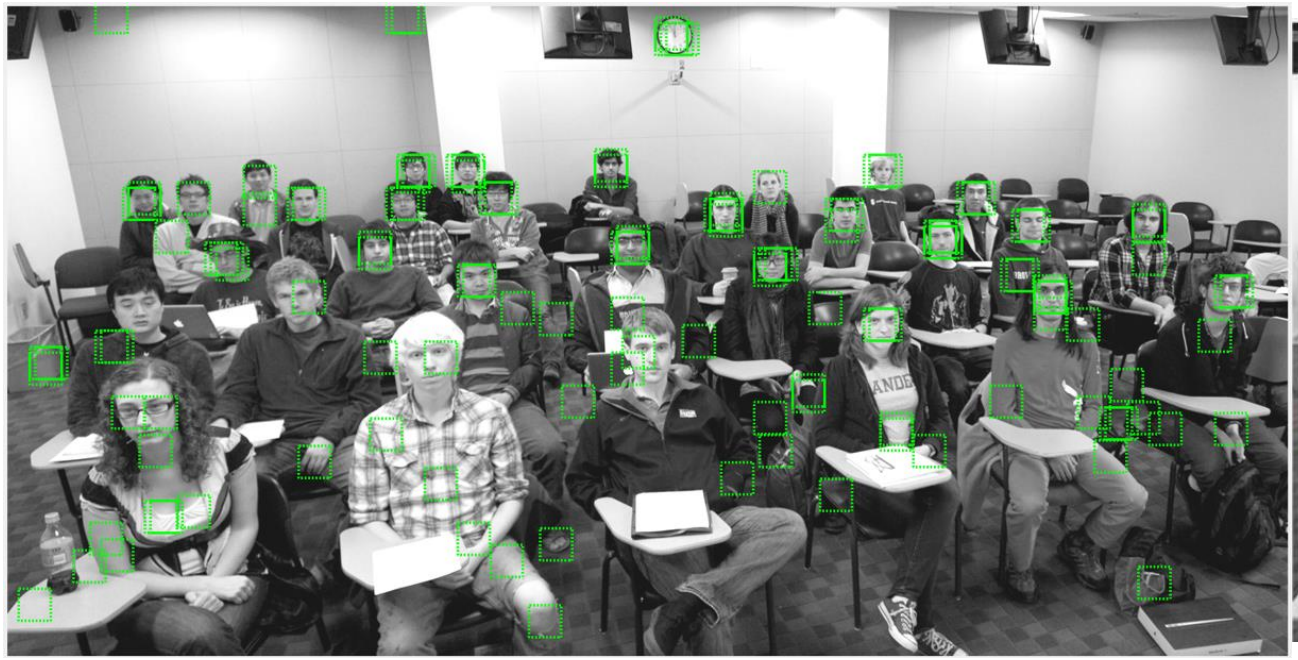
图片以及未经过NMS的候选框



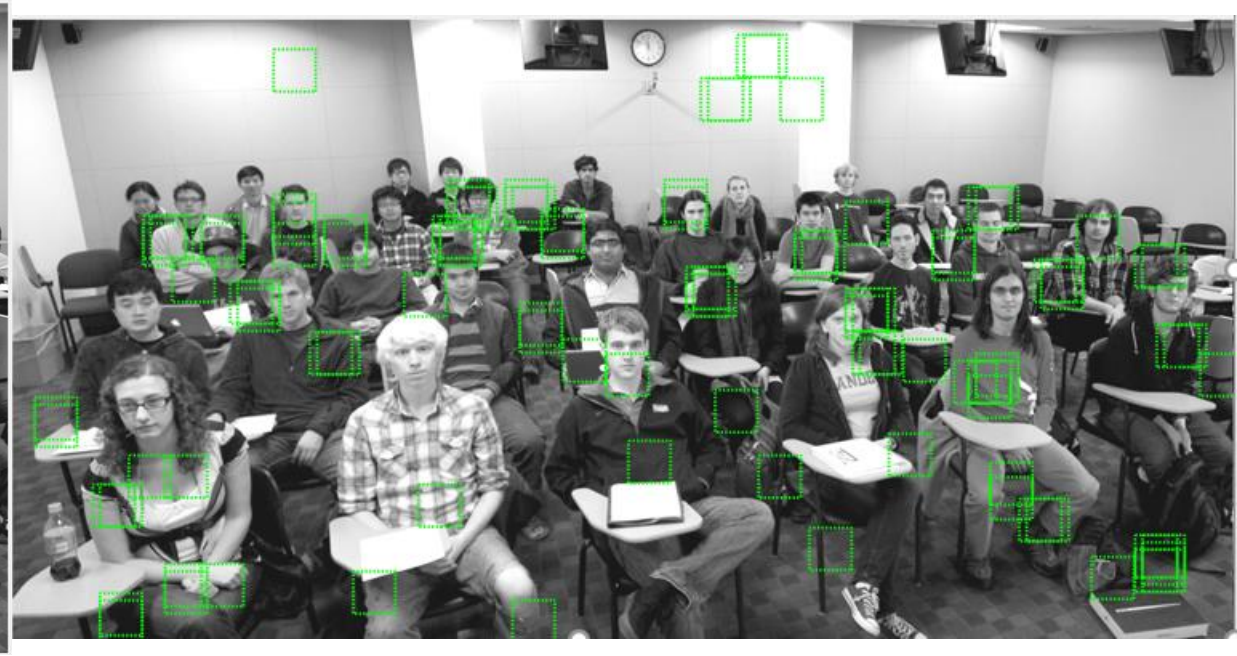
可视化HOG特征



Project 4: Face Detection with a sliding window



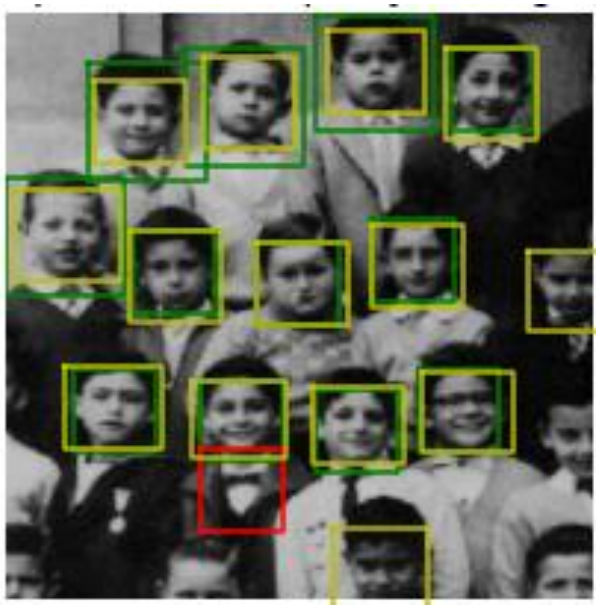
0.7scale的图片

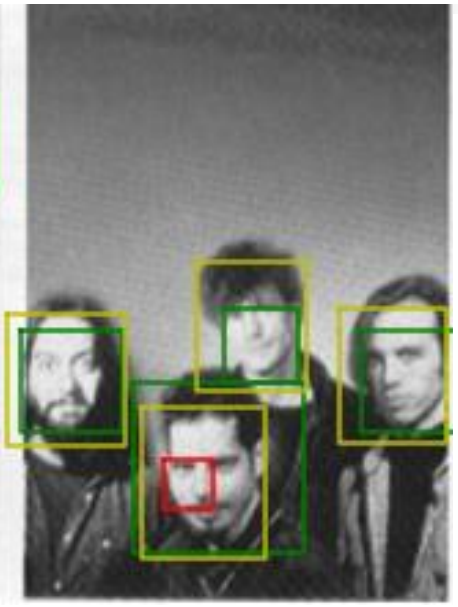


原尺寸的图片

Project 4: Face Detection with a sliding window

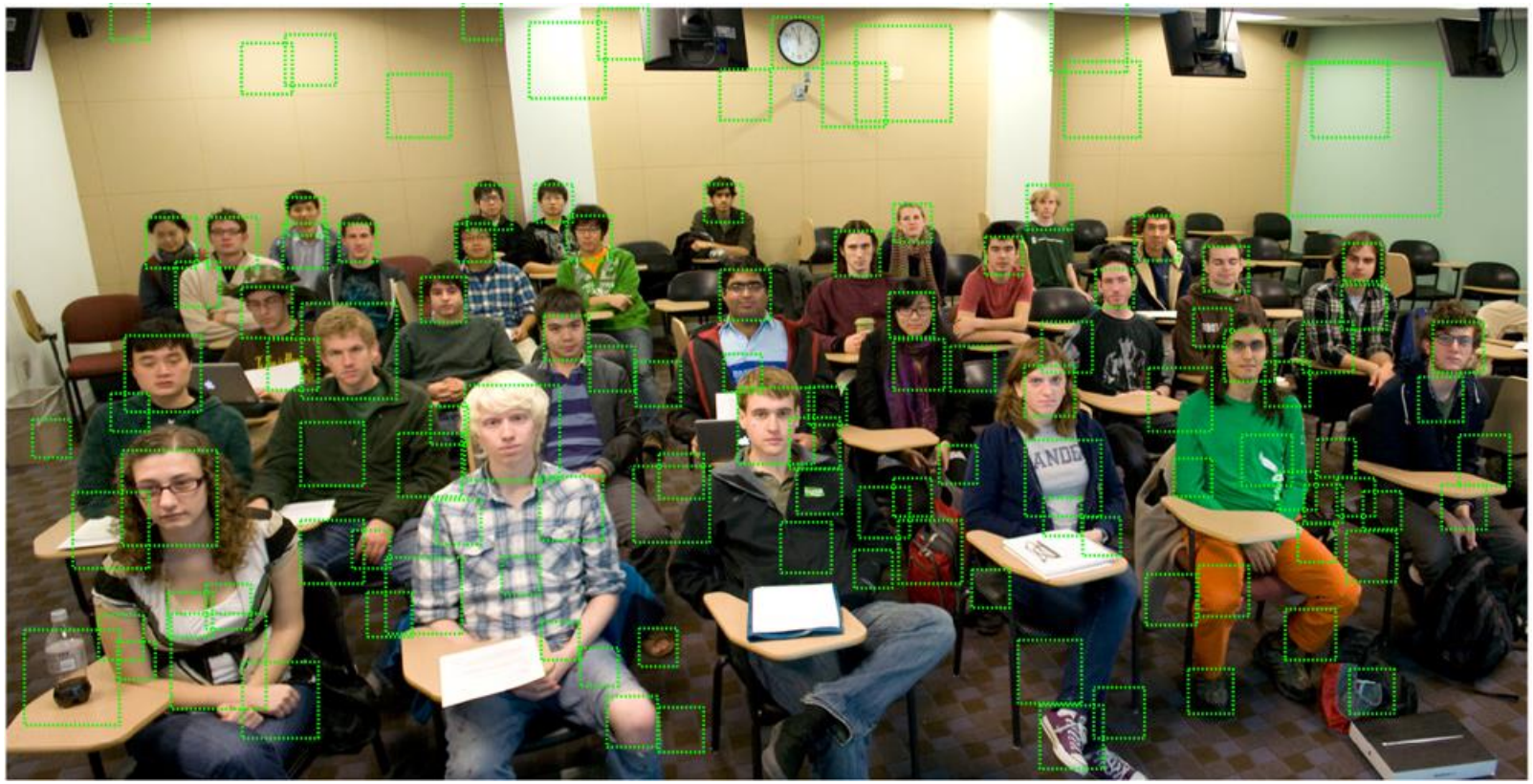
查看模型效果





Project 4: Face Detection with a sliding window

在额外的图片上测试（optional）



Project 5: Fish Detection with Deep Learning

目的：

基于YOLOv3-tiny模型实现🐟检测算法。熟悉深度学习在目标检测上的应用方式。

操作关键点：

1. 数据集划分

   图片和标注已经提供，编写代码进行**训练集**和**验证集**的划分

2. 编写训练阶段的代码

代码：

一共有两处，全部在proj5.ipynb中

Project 5: Fish Detection with Deep Learning

Data Preprocess

You should code this part first

```
[-]
    ##########################################################################################################
    #                                          Your Code                                                     #
    ##########################################################################################################
    # You should generate valid Train dataset and Val dataset.
    # Use data in data/custom/images and data/custom/labels to generate the path file train.txt and
    # val.txt in data/custom/
    # a qualified val dataset is smaller than the train dataset and
    # most time there are no overlapped data between two sets.









    ##########################################################################################################
    #                                            End                                                         #
    ##########################################################################################################
```

Train your model!

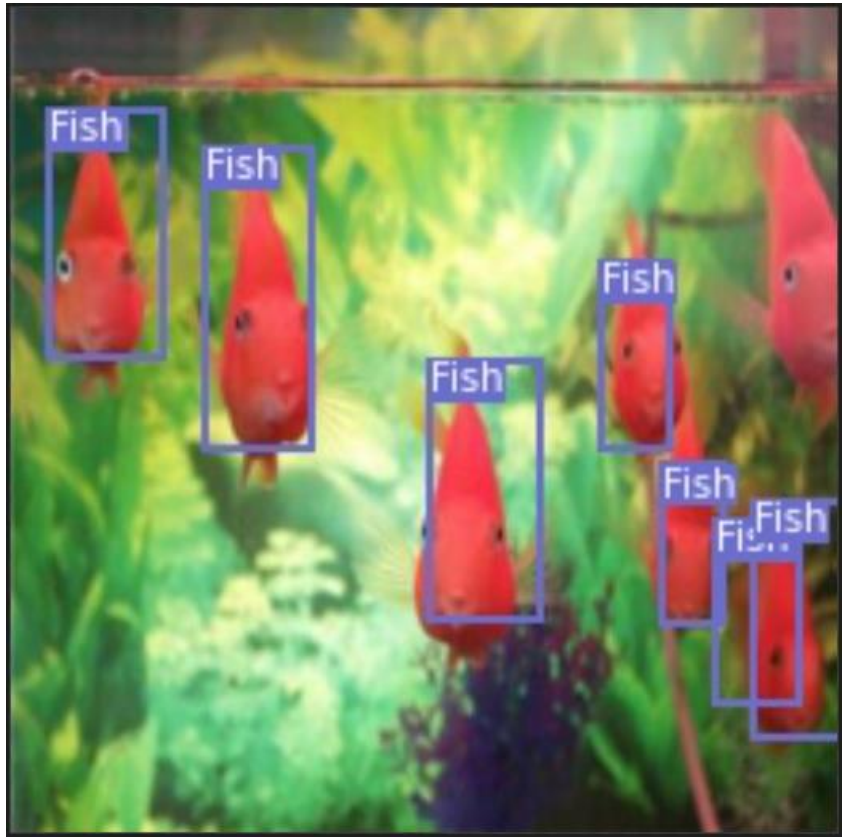
You are required to complete the DL project training steps (get data batch from dataloader, forward, compute the loss
comments.

```
[-]
    for epoch in range(opt["epochs"]):
        print("\n---- Training Model ----")
        model.train()
        ##########################################################################################################
        #                                          Your Code                                                     #
        ##########################################################################################################
        # Your code need to execute forward and backward steps.
        # Use 'enumerate' to get a batch[_, images, targets]
        # some helpful function
        # - outputs = model.__call__(imgs)(use it by model(imgs))
        # - loss, _ = cumpte_loss(outputs, targets, model)
        # - loss.backward() (backward step)
        # - optimizer.step() (execute params updating)
        # - optimizer.zero_grad() (reset gradients)
        # if you want to see how loss changes in each mini-batch step:
        # -eg print(f'Epoch:{epoch+1}, Step{step+1}/{len(dataloader)}, loss:{loss.item()}')









        ##########################################################################################################
        #                                            End                                                         #
        ##########################################################################################################
```

Project 5: Fish Detection with Deep Learning

```
---- Training Model ----
Epoch:46, Step1/1, loss:0.24913674592971802

---- Training Model ----
Epoch:47, Step1/1, loss:0.2667277157306671

---- Training Model ----
Epoch:48, Step1/1, loss:0.2737748920917511

---- Training Model ----
Epoch:49, Step1/1, loss:0.29579898715019226

---- Training Model ----
Epoch:50, Step1/1, loss:0.5155555605888367
```

```
---- Evaluating Model ----
Detecting objects: 100%|          | 1/1 [00:02<00:00,  2.27s/it]
Computing AP: 100%|          | 1/1 [00:00<?, ?it/s]['Fish'] 0
+-------+------------+---------+
| Index | Class name | AP      |
+-------+------------+---------+
| 0     | Fish       | 0.95238 |
+-------+------------+---------+
---- mAP 0.9523809523809523
```



Thank You!

未来媒体研究中心
CENTER FOR FUTURE MEDIA

电子科技大学
University of Electronic Science and Technology of China