计算机视觉，2022 春季学期

编程作业 4

基于词袋模型的场景识别

说明

1. 诚信与协作：鼓励学生分组工作，但每个学生都必须提交自己的作品。如果你作为一个团队工作，请在你的文章中包含你的合作者的姓名。不应共享或复制代码。除非允许，否则请不要使用外部代码。抄袭被强烈禁止，并可能导致本课程失败。
2. 尽早开始！尤其是那些不熟悉 *Python* 的人。在 *Q*2.3 中构建字典，以及在 *Q*3.1 中将所有图像转换为视觉单词，运行起来可能需要长达 60 分钟（如果实施得好！）。如果你开始晚了，你在调试的时候会被赶时间。
3. 问题：如果你有任何问题，请先查看 *Piazza*。其他同学可能也遇到过同样的问题，而且可能已经解决了。如果没有，请在讨论板上发布你的问题。*TA* 会尽快回复。
4. 写作：每个问题都提到了写作中要包含的项目，并在“写作”部分进行了总结。请注意，我们不接受你在此作业中撰写的手写扫描件。请以电子方式输入你对理论问题和实验讨论的答案。
5. 讲义：讲义 *zip* 文件包含 3 个项目。*assgn*4*.pdf* 是作业讲义。*data/*包含来自 *SUN* 图像数据库的 1491 个图像文件，分为 8 个类别文件夹。它还包含一个 *Python pickle* 文件 *traintest.pkl*，其中包含所有训练和测试图像路径以及标签。*python/*包含你将在此项目中使用的脚本。
6. 代码：坚持讲义中提到的函数原型。这使得 *TA* 更容易验证代码。如果你确实想更改函数原型或添加额外参数，请与助教联系。
7. *Python*：本作业基于 *Python* 3。你将需要使用以下软件包：*numpy*、*scipy*、*pickle*、*sklearn*、*skimage*、*opencv*。
8. 提交：你提交的这个作业应该是一个 *zip* 文件，<*andrew-id.zip*>，由你的文章、你的 *Python* 实现（包括辅助函数）和你的实现、额外学分的结果（可选）组成。不要交出我们在讲义 *zip* 中分发的图像文件，或任何生成的字图文件。但是，你应该提交你生成的字典和视觉*.pkl* 文件。

9. 你最终上传的文件应按此布局排列：

- <用户名+学号>.*zip*
  - <用户名+学号>.*pdf*
  - *ec/*
    - *computeIDF.py* (*Q*5.2)
    - *evaluateRecognitionSystem IDF.py* (*Q*5.2)
    - *evaluateRecognitionSystem SVM.py* (*Q*5.1)
    - *tryBetterFeatures.py* (*Q*5.3)
    - 任何其他辅助函数或外部代码
  - *python/*
    - *RGB2Lab.py* (已提供)
    - *batchToVisualWords.py* (已提供)
    - *buildRecognitionSystem.py* (*Q*3.3)
    - *createFilterBank.py* (已提供)
    - *dictionaryHarris.pkl* (*Q*2.3)
    - *dictionaryRandom.pkl* (*Q*2.3)
    - *evaluateRecognitionSystem NN.py* (*Q*4.2)
    - *evaluateRecognitionSystem kNN.py* (*Q*4.2)
    - *extractFilterResponses.py* (*Q*2.1)
    - *getDictionary.py* (*Q*2.3)
    - *getHarrisPoints.py* (*Q*2.2)
    - *getImageDistance.py* (*Q*4.1)
    - *getImageFeatures.py* (*Q*3.2)
    - *getRandomPoints.py* (*Q*2.2)
    - *getVisualWords.py* (*Q*3.1)
    - *utils.py* (已提供)
    - *visionHarris.pkl* (*Q*3.3)
    - *visionRandom.pkl* (*Q*3.3)
    - 任何其他辅助函数

1 概述

计算机视觉的核心问题之一是分类。给定一张来自几个固定类别的图像，你能确定它属于哪个类别吗？对于这项任务，你将开发一个场景分类系统。拥有大量用户上传照片的服务（如 *Flickr* 或 *Instagram*）可能会使用这样的系统，这些服务希望根据风景类型自动对照片进行分类。机器人系统也可以使用它来确定它所处的环境类型，并可能改变它的移动方式。

你已获得由八个场景类别组成的 *SUN Image* 数据库子集。请参见图 1。在此任务中，你将构建一个端到端系统，该系统将在给定新场景图像的情况下确定它是哪种类型的场景。

图 1：你从 *SUN Image* 数据库中获得的八个类别

该作业基于一种称为“*Bag of Words*”（词袋）的文档分类方法。这种文档分类方法将文档表示为文档中出现单词的词向量或计数直方图，如图 2 所示。希望同一类中的不同文档具有相似的单词集合和分布，并且当我们看到一个新文档时，我们可以通过将其与该类中已有的直方图进行比较来找出它属于哪个类。这种方法在自然语言处理中非常成功，由于其相对简单而令人惊讶。我们将采用相同的方法进行图像分类。然而，出现了一个主要问题。对于文本文档，我们实际上有一个单词字典。但是我们可以用什么词来表示图像呢？

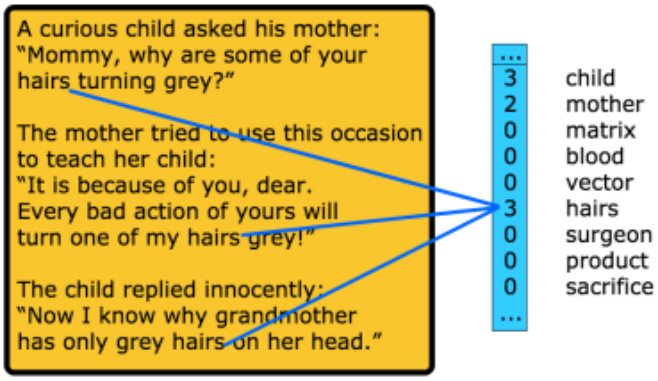

图 2：文本文档的词袋表示

该作业有三个主要部分：第 2 节涉及从训练数据中构建视觉单词词典；第 3 节涉及使用我们的视觉单词典和训练图像构建识别系统；在第 4 节中，你将使用测试图像评估识别系统。

在第 2 节中，你将使用提供的滤波器组将每个图像的每个像素转换为高维表示，该表示将捕获有意义的信息，例如角、边缘等…… 该操作将使每个像素从颜色的 3 维向量，转换到滤波器响应下的 n 维向量。然后，从所有训练图像中获取这些 n 维像素，并运行 *K-means* 方法进行聚类。每个生成的聚类中心将成为一个视觉单词，所有聚类中心组合在一起就是我们的视觉单词词典。理论上，我们希望使用所有训练图像中的所有像素，但这在计算上非常昂贵。相反，我们只会从每个图像中抽取一小部分像素样本。一种选择是简单地从每个像素中随机均匀地选择 α 个像素。另一种选择是使用一些特征检测器（例如 *Harris* 角点检测器），并从每个图像中获取 α 个特征点。你将同时制作两本词典，以便我们比较它们的相对性能。请参见图 3。

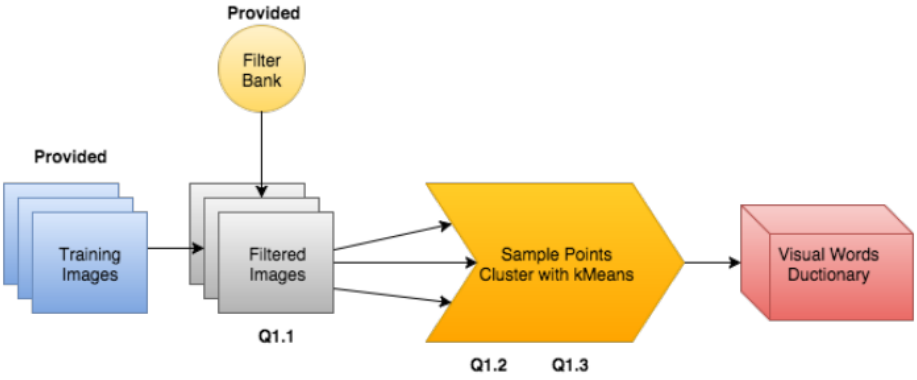

图 3：项目第一部分的流程图，涉及构建视觉单词词典

在第 3 节中，你生成的视觉单词词典将应用于每个训练图像，以将它们转换为**视觉单词的映射图**（*word map*）。该操作会为过滤后的图片上每一个像素分配一个整数标签，标签与词典的聚类中心编号一一对应。然后每张图片都会被转换成一个“词袋”；视觉单词计数的直方图。然后，你将使用这些来构建分类器（最近邻和 *SVM* 以获得额外的分数）。参见图 4。

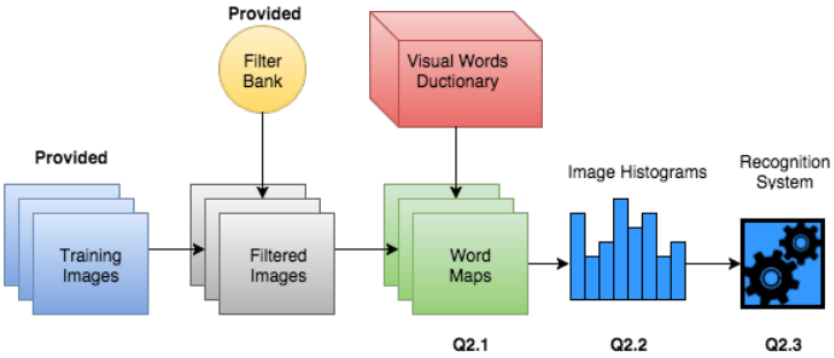

图 4：涉及构建识别系统的项目第二部分的流程图

在第 4 节中，你将评估你构建的识别系统。这将涉及获取测试图像并使用视觉单词汇词典和你在第 3 节中编写的函数将它们转换为图像直方图。接下来，对于最近邻分类，你将使用直方图距离函数将新的测试图像直方图与训练图像直方图以便对新的测试图像进行分类。对所有测试图像执行此操作将使你了解你的识别系统有多好。请参见图 5。

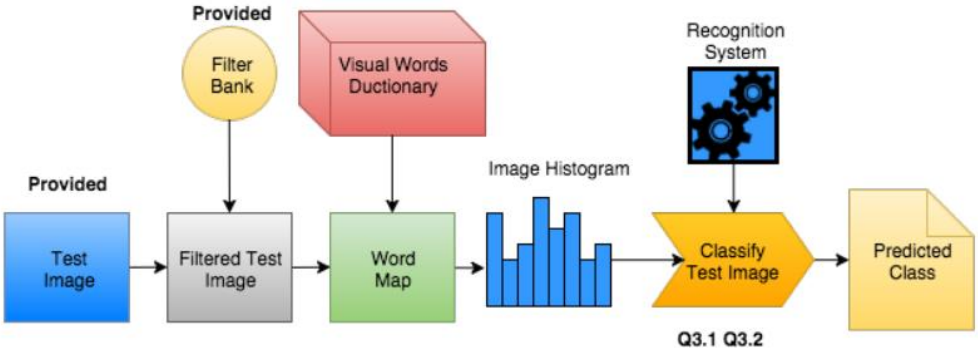

图 5：涉及评估识别系统的项目第三部分的流程图

2 建立视觉单词汇词典

Q2.1 提取滤波器响应（20 分）

编写一个函数来提取滤波器响应。

filterResponses = extract_filter_responses(*I*, *filterBank*)

我们提供了函数 *createFilterBank*()。该函数将生成一组图像卷积滤波器。参见图 6。有 4 个滤波器，每个滤波器按 5 个不同的比例制作，总共 20 个滤波器（5*x*4=20）。滤波器是：

- 高斯：对恒定区域响应强烈，并抑制边缘、角落和噪声
- 高斯拉普拉斯算子：对与滤波器大小相似的斑点反应强烈
- 高斯 *X* 梯度：对垂直边缘反应强烈
- 高斯 *Y* 梯度：对水平边缘反应强烈

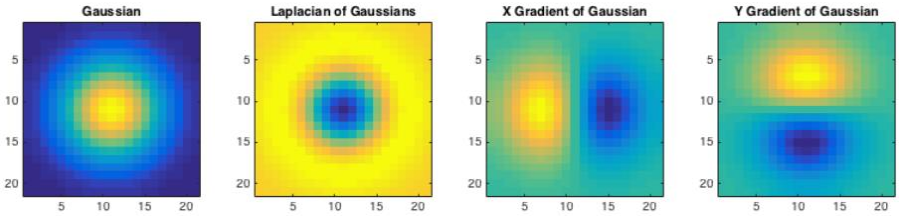

图 6：我们为你提供的四种类型的滤波器

将此图像卷积滤波器数组传递给你的 *extract_filter_responses*（..）函数，以及大小为 *H*×*W*×*3* 的图像。你的函数应使用提供的 *rgb2Lab*(..)函数将 *I* 的颜色空间从 *RGB* 到 *Lab*。然后它应该在输入图像的 3 个颜色通道中的每一个上应用所有 *n* 个滤波器。你应该最终得到图像的 3*n* 个滤波器响应。对于图像过滤，你可以使用 *utils.py* 中提供的辅助函数 *imfilter*(..)。返回的最终矩阵的大小应为 *H*×*W*×3*n*（图 7）。

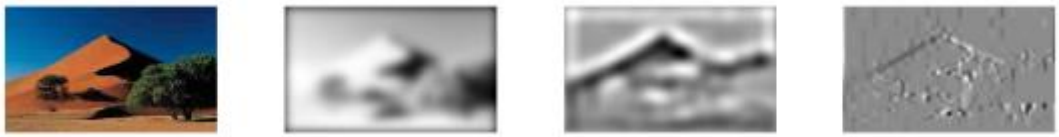

图 7：沙漠图像示例和一些滤波器响应（在 *CIE Lab* 颜色空间中）

在你的文章中：显示数据集中的图像及其 3 个滤波器响应。解释你可能注意到的任何工件。另外，简要描述 *CIE Lab* 色彩空间，以及我们为什么要使用它。我们没有在课堂上介绍 *CIE Lab* 色彩空间，因此你需要在线查找。

Q2.2　从图像中收集点样本（20 分）

编写两个函数，返回图像中的像素点列表，然后将其用于生成视觉单词。

首先，编写一个简单的函数，它接收一张图像和一个 α，并返回一个大小为 α×2 的矩阵，其中包含图像内的随机像素位置。

points = get_random_points(I, alpha).

接下来，编写一个函数，使用 *Harris* 角点检测算法从输入图像中选择关键点，使用课堂中所示的类似算法。

points = get_harris_points(I, alpha, k)

回想一下课堂上讲过的，*Harris* 角点检测器通过在图像中的点周围的区域内构建边缘梯度的协方差矩阵来找到角点。该矩阵的特征向量指向变化最大的两个方向。如果它们都很大，那么这表示一个角点（*corner*）。有关更多详细信息，请参阅课程幻灯片。

该函数取输入图像 I。I 可能是彩色图像或灰度图像，但应对图像的灰度表示进行以下操作。对于每个像素，你需要计算协方差矩阵

$$M = \begin{bmatrix} \sum_{p \in P} I_{xx} & \sum_{p \in P} I_{xy} \\ \sum_{p \in P} I_{yx} & \sum_{p \in P} I_{yy} \end{bmatrix}$$

其中 $I_{ab} = \frac{\partial I}{\partial a}\frac{\partial I}{\partial b}$，$P$ 是像素的邻域。P 可以使用 3×3 或 5×5。最好预先计算图像的 X 和 Y 梯度，而不是在循环的每次迭代中都进行计算。总而言之，还要考虑如何使用卷积滤波器来做到这一点。

然后，你希望通过查找协方差矩阵特征值较大的像素来检测角点。由于显式计算特征值的成本很高，因此你应该使用

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k\ tr(M)^2$$

其中，*det* 表示行列式，*tr* 表示矩阵的迹。回想一下，当使用 *Harris* 角点检测器检测角点时，角点是两个特征值都很大的点。这与边缘（一个特征值很大，而另一个很小）和平坦区域（两个特征值都很小）形成对比。在响应函数中，如果其中一个特征值很小，则第一项变得非常小，从而使 R<0。R 值越大表示特征值也同样大。

无需对响应函数进行阈值处理，只需将前 α 个响应作为角点，并返回它们的坐标。k 参数的参考范围是 0.04-0.06。

注意：这个 Harris 可以在没有任何 for 循环的情况下实现，使用基本的 numpy 函数、过滤和向量化，

特别是计算 M 中的和，以及响应函数 R。对于慢代码不会取消标记，但是在 Q2.3 中计算视觉单词词典时，这可能是分钟和小时之间的差异。请记住，你将将此功能应用于大约 1500 张图像。

这些函数将在下一部分中用于从每个训练图像中选择 α 个点。

在你的文章中：在 3 个不同的图像上显示角检测器的结果。

图 8：*getPoints* 函数的可能结果

Q2.3　计算视觉单词词典（20 分）

你现在将创建视觉单词词典。编写函数：

dictionary = *get_dictionary*(*imgPaths*, *alpha*, *K*, *method*)

此函数接受一组训练图像路径。对于每个训练图像，加载它并对其应用滤波器组。然后为每个图像获取 α 个点并将它们放入一个数组中，其中每一行代表一个 n 维像素（n 是滤波器的数量）。如果总共有 T 个训练图像，那么你将构建一个大小为 $\alpha T \times 3n$ 的矩阵，其中每一行对应一个像素，并且总共收集了 αT 个像素。

方法是从“*Random*”或“*Harris*”中选择，这将确定如何从每个图像中获取 α 点。如果方法是“*Random*”，那么将使用 *get_random_points*(..)来选择 α 点，而“*Harris*”方法将告诉你的函数使用 *get_harris_points*(..)来选择 α 点。完成此操作后，将从所有训练图像中收集的所有点的像素响应传递给 *sklearn* 的 *K-means* 函数以获取字典。

结果将是一个大小为 $K \times 3n$ 的矩阵，其中每一行代表一个聚类中心的坐标。这个矩阵将是你的视觉单词汇词典。

出于测试目的，使用 α=50 和 K=100。最终，你将希望使用更大的数字（例如，α=200 和 K=500）以获得更好的最后部分的结果。

此功能可能需要一段时间才能运行。尽早开始并耐心等待。完成后，你将拥有视觉单词汇词典。将其保存在 *pickle*(*.pkl*)文件中。这是你的视觉单词汇词典。编写一个 *computeDictionary.py* 可能会有所帮助，它将完成加载训练图像路径、处理它们、构建字典和保存 *pickle* 文件的工作。这不是必需的，但在多次调用时会对你有所帮助。

对于这个问题，你必须生成两本词典。一个名为 *dictionaryRandom.pkl*，它使用随机方法来选择点，另一个名为 *dictionaryHarris.pkl*，它使用 *Harris* 方法。两者都必须上交。

3 搭建视觉场景识别系统

Q3.1 将图像转换为词图（30 分）

编写一个函数，将图像中的每个像素映射到字典中最接近的单词。

wordMap = *get_visual_words*(*I*, *filterBank*, *dictionary*)

I 是大小为 *H*×*W*×3 的输入图像。*dictionary* 是先前计算的字典。*filterBank* 是用于构造字典的滤波器库。*wordMap* 是一个 *H*×*W* 整数标签矩阵，其中每个标签对应于字典中的一个词/簇中心。

使用 *scipy* 的函数 *cdist*(..)的 *euclidean* 距离来有效地做到这一点。你可以使用 *skimage.color* 函数 *label2rgb*(..)可视化你的结果。

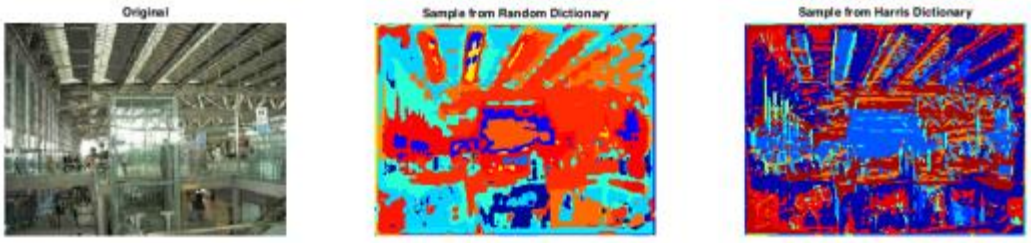

图 9：示例视觉单词图，使用 α=50 和 K=50 从字典制作

完成后，调用提供的脚本 *batchToVisualWords.py*。此函数会将你的 *get_visual_words*(..)实现应用到训练和测试集中的每个图像。该脚本将加载 *traintest.pkl*，其中包含所有数据图像的名称和标签。对于每个训练图像 *data/<category>/X.jpg*，这个脚本会保存 *word_map_pickle file data/<category>/X_<point_method>.pkl*。为获得最佳速度，请将 *num_cores* 修改为计算机中的内核数。除非你决定更改字典，否则这将使你不必在后面的部分中继续重新运行 *get_visual_words*。

在你的文章中：显示来自两个不同类别的 3 个不同图像的单词映射（总共 6 个图像）。对两种字典类型（随机和哈里斯）中的每一种都执行此操作。视觉单词是否捕捉语义？在你看来，哪本词典似乎更好？为什么？

Q3.2　获取图像特征（10 分）

创建一个函数，用于提取给定图像（即视觉单词袋）中视觉单词的直方图。

h = *get_image_features*(*wordMap*, *dictionarySize*)

h，图像的向量表示，是大小为 1×*K* 的直方图，其中 *dictionarySize* 是字典中簇 *K* 的数量。*h*(*i*)应该等于视觉单词 *i* 在词图中出现的次数。

由于图像的大小不同，因此 *h* 中的总计数会因图像而异。为了解决这个问题，L_1 在从函数返回直方图之前对其进行归一化。

Q3.3　建立识别系统 - 最近邻（10 分）

既然你已经构建了一种获取输入图像的矢量表示的方法，那么你就可以构建视觉场景分类系统了。该分类器将使用最近邻分类。编写脚本 *buildRecognitionSystem.py* 保存 *visionRandom.pkl* 和 *visionHarris.pkl*，并在每个 *pickle* 中存储一个包含以下内容的字典：

1. *dictionary*：你的视觉单词词典，一个大小为 *K*×3*n* 的矩阵。
2. *filterBank*：滤波器库，用于产生字典。这是一组图像滤波器。
3. *trainFeatures*：*T*×*K* 矩阵，包含数据集中 *T* 个训练图像的所有视觉单词的直方图。
4. *trainLabels*：*T*×1 向量，包含每个训练图像的标签。

你需要从 *traintest.pkl* 加载 *train_imagenames* 和 *train_labels*。从你在 *Q*2.3 部分中保存的 *dictionaryRandom.pkl* 和 *dictionaryHarris.pkl* 加载词典。

4 评估视觉场景识别系统

Q4.1 图像特征距离（15 分）

对于最近邻分类，你需要一个函数来计算两个图像特征向量之间的距离。写一个函数

dist = *get_image_distance*(*hist*1, *hist*2, *method*)

*hist*1 和 *hist*2 是两个图像直方图，将计算两者距离（以 *hist*2 为目标），并在 *dist* 中返回。这个想法源于两个非常相似的图像应该有一个非常小的距离，而不同的图像应该有一个更大的距离。

方法将控制如何计算距离，并将设置为"*euclidean*"或"*chi*2"。第一个选项告诉计算两个直方图之间的欧几里得距离。第二个使用χ^2距离（卡方距离）。对于χ^2距离（卡方距离），你可以使用 *utils.py* 中的函数 *chi*2*dist*(..)。

或者，你也可以编写函数

[*dist*] = *get image distance*(*hist*1, *histSet*, *method*)

其中，不采用第二个直方图，它采用直方图矩阵，并返回 *hist*1 和 *histSet* 中每个直方图之间的距离向量。这可能使更有效地实现事物成为可能。当然，你需要修改 *chi*2*dist*(..)来处理多个距离的计算。选择一个或另一个上交。

Q4.2 评估识别系统-*NN* 和 *kNN*（40 分）

1. *evaluateRecognitionSystem_NN.py*

编写脚本 *evaluateRecognitionSystem_NN.py* 在测试图像上评估你的最近邻识别系统。最近邻分类器为测试图像分配了与训练集中"*Nearest*"样本相同的类别。"*Nearest*"由你的距离函数定义。

加载 *traintest.pkl* 并对每个 *test_imagenames* 文件进行分类。让脚本报告准确度($\frac{\#correct}{\#total}$)以及 8×8 混淆矩阵 *C*，其中 *C*(*i*,*j*)记录实际类别 *i* 的图像被分类为类别 *j* 的次数。

混淆矩阵可以帮助你确定哪些类比其他类更有效，并在每个类的基础上量化你的结果。在一个完美的分类器中，你只会在 *C* 的对角线上有条目（例如，这意味着"*auditorium*"总是被正确分类为"*auditorium*"）。

对于字典（*Random* 或 *Harris*）和距离度量（*Euclidean* 和χ^2）的每种组合，让你的脚本打印出混淆度量和混淆矩阵。使用 *print*(..)这样我们就可以知道哪个是哪个。

2. *evaluateRecognitionSystem_kNN.py*

现在采用字典和距离度量的最佳组合，并编写脚本 *evaluateRecognitionSystem_kNN.py* 使用 *k Nearest Neighbors* 对所有测试图像进行分类。让你的脚本生成从 1 到 40 的 *k* 精度图。要获得最佳性能 *k* 值，请打印混淆矩阵。请注意，此 *k* 与字典大小 *K* 不同。此 *k* 是对新测试图像进行分类时要考虑的附近点的数量。

在你的文章中：

- 包括 *evaluateRecognitionSystem_NN.py* 的输出（4 个混淆矩阵和准确度）。
- 两本词典的表现如何比较？这令人惊讶吗？
- 这两个距离度量如何影响结果？哪个表现更好？你认为这是为什么？
- 还包括 *evaluateRecognionSystem kNN.py* 的输出（绘图和混淆矩阵）。评论 *k* 的最佳值。*k* 越大越好吗？为什么或者为什么不？你是如何选择解决关系的？

5 额外加分

Q5.1 评估识别系统 - 支持向量机（20 分）

在课堂上，我们通过名称支持向量机(*SVM*)了解了一个强大的分类器。编写一个脚本 *evaluateRecognitionSystem_SVM.py* 来使用 *SVM* 进行分类。生成一个新的 *visionSVM.pkl* 文件，其中包含你需要的任何参数，并在测试集上对其进行评估。

我们建议你将 *sklearn*(*https://scikit-learn.org/stable/modules/svm*)用于 *SVM*。尝试至少两种类型的内核。

在你的文章中： *SVM* 的性能是否优于最近邻？为什么或者为什么不？一个内核是否比另一个内核工作得更好？为什么？

Q5.2 逆文档频率（10 分）

使用词袋模型，图像识别类似于用词对文档进行分类。在文档分类中，结合了逆文档频率（*IDF*）因子，它减少了文档集中出现频率很高的术语的权重。例如，因为"*the*"这个词很常见，这往往会错误地强调那些碰巧更频繁地使用"*the*"这个词的文档，而没有对更有意义的词给予足够的重视。

在这个项目中，我们计算的直方图只考虑了词频（*TF*），即该词在词图中出现的次数。现在我们想通过它的逆文档频率来加权这个词。一个词的 *IDF* 定义为：

$$IDF_w = \log \frac{T}{|\{d: w \in d\}|}$$

这里，T 是所有训练图像的数量，$|\{d: w \in d\}|$是含有单词 w 的图像的数量。

编写脚本 *computeIDF.py* 来计算大小为 $1 \times K$ 的向量 *IDF*，其中包含所有视觉单词的 *IDF*，其中 K 是字典大小。将 *IDF* 保存在 *idf.pkl* 中。然后编写另一个脚本 *evaluateRecognitionSystem IDF.py* 在识别过程中使用 *IDF* 向量。你可以使用最近邻或 *SVM* 作为分类器。

在你的文章中： 逆文档频率如何影响性能？更好或更差？这有意义吗？

Q5.3　更好的像素特征（20 分）

我们提供给你的滤波器组包含一些简单的图像滤波器，例如高斯、导数和拉普拉斯算子。然而，我们在课堂上了解了其他各种方法，例如 *Haar wavelets*、*Gabor filters*、*SIFT*、*HOG* 等……尝试任何你认为可行的方法。包括脚本 *tryBetterFeatures.py* 和你需要的任何其他代码。随意使用你在网上找到的任何代码或包，但一定要清楚地引用它。尝试任何你认为合适的东西，只要确保提交不会变得太大。

在你的文章中：你做了什么实验以及它的表现如何。准确度是多少？

6 写作总结

Q2.1 显示数据集中的图像及其滤波器响应的 3 个。解释你可能注意到的任何工件。还简要描述了 *CIE Lab* 色彩空间，以及我们为什么要使用它。我们没有在课堂上介绍 *CIE Lab* 色彩空间，因此你需要在线查找。

Q2.2 在 3 张不同的图像上显示角检测器的结果。

Q3.1 显示来自两个不同类别的 3 个不同图像的单词图（总共 6 个图像）。对两种字典类型（*Random* 或 *Harris*）中的每一种都执行此操作。视觉单词是否捕捉语义？在你看来，哪本词典似乎更好？为什么？

Q4.1 对整个系统的评论：

- 包括 *evaluateRecognitionSystem NN.py* 的输出（4 个混淆矩阵和准确度）。
- 两本词典的表现如何比较？这令人惊讶吗？
- 两个距离指标怎么样？哪个表现更好？你为什么认为这是？
- 还包括 *evaluateRecognionSystem kNN.py* 的输出（绘图和混淆矩阵）。评论 k 的最佳值。k 越大越好吗？为什么或者为什么不？你是如何选择解决关系的？

Q5.1 （额外分数）使用 *SVM* 分类获得的性能是否比使用最近邻的更好？为什么或者为什么不？ 一个内核是否比另一个内核工作得更好？为什么？

Q5.2 （额外分数）逆文档频率如何影响性能？更好或更差？这有意义吗？

Q5.3 （额外分数）你做了什么实验以及它的表现如何。准确度是多少？

参考文献

[1] *S. Lazebnik, C. Schmid, and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. CVPR*, 2006