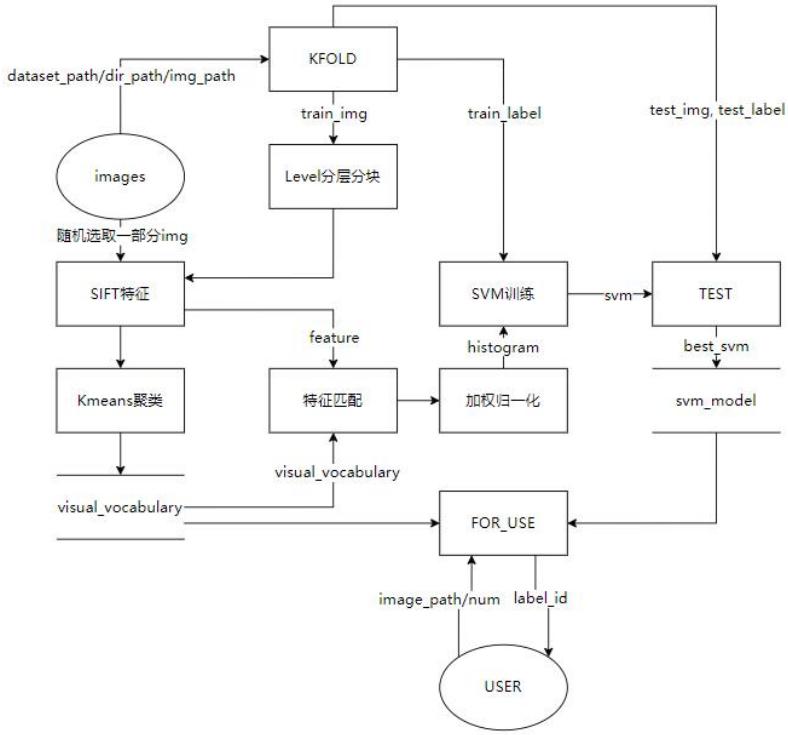
一、算法流程

1、数据流图



2、算法流程文字描述

2.1 视觉词汇表建立

2.1.1 选取所有图像/随机选取图像

2.1.2 计算 SIFT 特征

2.1.3 对特征做 KMEANS 聚类，聚类中心作为视觉词汇

2.1.4 保存词汇表以供使用

2.2 分层进行匹配得到图像表示

2.2.1 分层分块计算 SIFT

2.2.2 计算与词汇表中词汇距离（欧式距离）

2.2.3 与距离最小（若干个）的词汇达成匹配

2.2.4 加权匹配并拼接直方图（层数越大，权重越大）

2.2.5 归一化

2.3 SVM 分类

2.3.1 使用 k 折交叉验证进行 SVM 分类

2.3.2 评估并保存模型参数以供使用

二、核心函数功能

1、从每个类别中随机抽取指定数量的图像并提取其 SIFT 特征

get_random_sift_features(data_dir, num_per_class=100)

参数	data_dir（str）：数据总路径。 num_per_class（int）：每类抽取的用于构建视觉词表的图像数。默认为 100。
返回值	np.array(all_features)（ndarray）：所有特征的 SIFT 描述符数组。用于词汇聚类。
注意	使用 random.sample(images, num_per_class)实现随机抽样；若类内图片不足 num_per_class 张，则选取全部。

2、计算指定层 SIFT 特征描述符 extract_sift_features_v2(image_path, layer_id)

参数	image_path（str）：图片路径。 layer_id（int）：当前层号（[0,L-1]）。
返回值	descriptors（list）：SIFT 描述符列表，最大长度为当前层图像的块数（(2**(layer_id))**2）。
注意	按维度从(0,0)开始等分，剩余边缘部分舍弃。

3、直方图表示的软分配索引项获取 get_closest_cluster_indices(distances, k=5, like=50)

参数	distances（list）：SIFT 特征与 K 个词汇之间的距离表。 k（int）：最多允许分配的词汇数。默认为 5。 like（int）：允许分配的词汇距离与最小距离的最大差值。默认为 50。
返回值	closest_indices（list）：可以平均分配的词汇项的索引。

结合 2、3，利用 bag_of_words_representation_v3(image_path, visual_vocabulary, L=2)函数逐层计算 SIFT 特征描述符、比对词汇距离、选择最小的 x（<=k）个词汇平均分配到直方图中并依次拼接，返回最终标准化的 histogram 用于 SVM 分类。

三、函数参数分析

1、视觉词汇数 K

默认为 200。一定范围内，K 越大，对图像描述越细粒度，分类效果越准确。

2、特征层数 L

取值大于等于 1，默认为 2。L 越大，特征表示越细粒度，越关注图像场景的局部信息。

四、实验结果

1、SIFT 提取示例

2、K 不变

（1）改变训练视觉词汇的图像数

①选取全部图像提取 SIFT 特征得到聚类中心，L=1：

```
Performing image classification...
Accuracy: 0.6142697881828316
Accuracy: 0.5819397993311036
Accuracy: 0.5997770345596433
Accuracy: 0.6031215161649944
Accuracy: 0.5719063545150501
Cross-validation scores: [0.6142697881828316, 0.5819397993311036, 0.599777034559
6433, 0.6031215161649944, 0.5719063545150501]
Mean accuracy: 0.5942028985507246
Saving model parameters...
```

②每类随机选取 100 张做聚类，L=1（略差于选取全部，但差别不大）：

	precision	recall	f1-score	support
0	0.46	0.46	0.46	46
1	0.75	0.92	0.83	52
2	0.33	0.23	0.27	56
3	0.37	0.24	0.29	42
4	0.54	0.47	0.50	75
5	0.50	0.65	0.56	74
6	0.74	0.93	0.82	57
7	0.70	0.58	0.63	57
8	0.65	0.63	0.64	65
9	0.53	0.72	0.61	57
10	0.73	0.43	0.54	81
11	0.42	0.56	0.48	54
12	0.73	0.49	0.59	75
13	0.48	0.67	0.56	45
14	0.48	0.51	0.49	61
accuracy			0.56	897
macro avg	0.56	0.57	0.55	897
weighted avg	0.57	0.56	0.56	897

```
Accuracy: 0.5641025641025641
Cross-validation scores: [0.5808249721293199, 0.5774804905239688, 0.602006688963
2107, 0.6020066889632107, 0.5641025641025641]
Mean accuracy: 0.5852842809364548
```

（2）直方图表示的分配方式与层数 L

在上例基础上，改变分配方式为平均软分配，L=1：

	precision	recall	f1-score	support
0	0.28	0.30	0.29	43
1	0.78	0.86	0.82	50
2	0.30	0.26	0.28	50
3	0.39	0.27	0.32	44
4	0.41	0.40	0.41	57
5	0.55	0.64	0.59	74
6	0.80	0.94	0.86	64
7	0.62	0.70	0.65	53
8	0.51	0.61	0.55	62
9	0.73	0.77	0.75	93
10	0.67	0.47	0.55	79
11	0.61	0.60	0.60	52
12	0.62	0.57	0.59	74
13	0.52	0.56	0.54	43
14	0.53	0.44	0.48	59
accuracy			0.58	897
macro avg	0.55	0.56	0.55	897
weighted avg	0.57	0.58	0.57	897

```
Cross-validation scores: [0.6098104793756968, 0.5919732441471572, 0.598662207357
8596, 0.6187290969899666, 0.5774804905239688]
Mean accuracy: 0.5993311036789297
```

有提升。

其他条件不变时，时间和平均准确率随 L 变化如下表：

Time/s / ACC/%	Avg_soft_5	Hard	Δ
L=1	586 / 59.93	504 / 58.24	82 / 1.69
L=2	1143 / 63.70	992 / 62.30	151 / 1.4
L=3	3704 / 63.95	3449 / 62.27	255 / 1.68

可以认为，在其他条件不变时，平均软分配带来的时间开销处于可接受范围内；对于 L 较小时对模型泛化能力改善效果最好；从 L=2 到 L=3 模型平均分类准确率提升不大。

（3）聚类算法

直接 200 聚类大约 3 小时，minibatch 聚类（初始默认 kmeans++）10s：

```
Creating visual vocabulary...
Build vocab time 10.052483797073364
Saving model parameters...
```

Minibatch 下 L=1 的软分配结果：

```
Cross-validation scores: [0.6086956521739131, 0.6042363433667781, 0.599777034559
6433, 0.6176142697881828, 0.5942028985507246]
Mean accuracy: 0.6049052396878484
```

时间成本显著降低，分类性能几乎无影响。

（4）分类算法

上述实验结果表明，当其他条件不变时（K=200，L=2，SOFT，Minbatch），使用单个 SVM 分类准确率 63.70%，若使用 5 个 SVM 组合的 Adaboost，此时分类准确率降低到 10%以下，考虑可能出现过拟合——在当前特征表示下，简单的分类器可以做出较好的效果。

（5）SVM 核

上述实验默认使用高斯核，若在其他条件不变情况下，使用线性核，此时分类准确率也降低到 10%以下。

3、L 不变

L=2，SOFT，Minibatch	Vocab_time/s	Data_time/s	Mean_acc/%
K=50	3.83	1000	59.44
K=100	32.85	1010	61.65
K=200（对照）	10.05	1143	63.70
K=300	16.71	1185	63.99
K=500	22.62	1535	64.59

可以进一步验证，Minibatch 不稳定（Vocab_time 与 K 值无明显相关关系）也表现在聚类时间上，但整体来讲较直接 kmeans 依旧能够显著降低词表构建时间成本，且对最终结果影响不大。

K 值在该条件下，与 Data_time 呈线性相关，但影响不是很大；在上述实验条件变化下，与准确率也呈线性相关。可以认为，本任务中，选取 K 在 200 及以上是合适的。

K 若继续扩大到 700/1000/1400（选取的用于构建词汇表的图像最大数），有如下结果：

L=2，SOFT，Minibatch	Vocab_time/s	Data_time/s	Mean_acc/%
K=700	35.97	1988	64.79
K=1000	51.77	3290	65.33
K=1400	74.88	4273	65.37

虽然准确率仍在提升，但速度放缓，时间代价也显著增加。综合来看 K=200 是合适的。

4、层级不做加权分配，只做简单累加

原来，我们为每个层级分配 1/(2**(L-1))的权重，当前层的特征是当前层检测到的特征减去更高一级的特征。而若我们不仔细做加权，只是简单地将每层检测到的特征做拼接，此时依旧满足层级高的特征在直方图中出现次数更多（因为在高层级出现的特征在低层级再次出现时会被重复统计），可以认为跟论文作者思路一致，但计算量相对较少。对比结果：

K=200，L=2，SOFT，Minbatch	Data_time/s	Mean_acc/%
原论文方法	1143	63.70
简单累加	1070	63.46

速度更快，准确率有所下降，但总体变化不大。

综上，在当前数据集下，考虑：每类随机选取 100 张构建词表，Minibatch K-means 做 K=200 聚类，层数 L=2，各层直方图软分配+简单累加拼接，SVM 选用高斯核（rbf）做图像分类。

	precision	recall	F1-score	accuracy
Macro avg	0.62	0.63	0.62	0.6388
Weight avg	0.64	0.64	0.63	

（注意，由于使用交叉验证筛选，没有按要求建议的数据集划分来实现模型训练。）