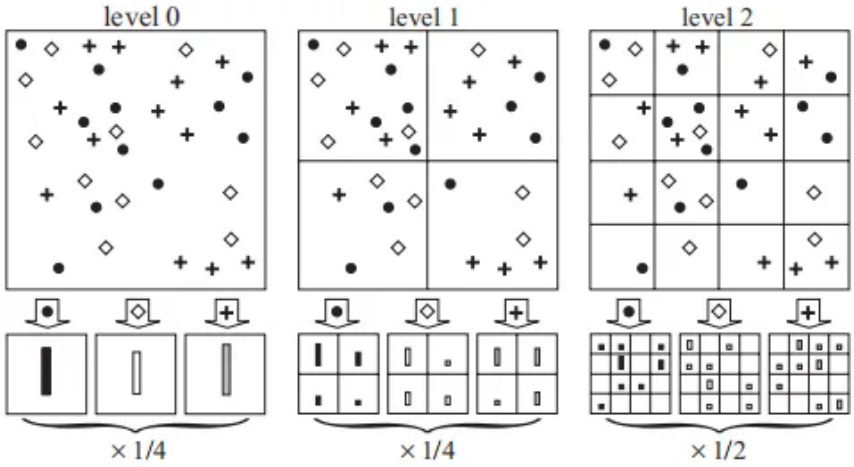
图像分类系统

班级：112班　姓名：吴晋彬　学号：2021213489

一、算法整体流程

1. 提取数据集中的样本，并划分训练集和测试集
2. 对于训练集和测试集的所有图片，提取图片的SIFT特征点，并对SIFT特征点向量归一化
3. 对训练集的所有SIFT特征点使用聚类算法分为n类，将n类特征点的中心点（即质心）作为视觉词汇，生成词袋。
4. SPM算法生成图片的特征向量



- 将图片分为3种尺度，分别为1*1、2*2、4*4大小，统计不同尺度下的特征直方图
- 将不同尺度下的特征直方图合并，组合成为一个[21,n]的特征向量

5. 将图片的特征向量作为训练数据，使用支持向量机分类
6. 预测结果
7. 评估预测结果，并生成分类报告和输出混淆矩阵

二、代码说明

（一）图像处理

(1) 图像特征提取

ImageProcess类，以下介绍类的属性和方法。

1. `class ImageProcess()`：储存图像的相关信息 `image,label` ,并对图像做SIFT特征提取

类的属性：

- image: 灰度图

- label: 图片的种类标签
- keypoints: 图片的所有特征点
- descriptors 图片的每个特征点所对应的向量 [len(keypoints),128]

2. `def __init__(self, image: np.ndarray ,label: int) -> None`

- 初始化

3. `def SIFT(self)`

- 提取图片的SIFT特征点,并对特征点向量归一化
- 调用 `cv2.SIFT_create().detectAndCompute`
- `descriptors` 每一行是一个点的特征向量，共有n行，取 `axis=1` 求每个特征向量的范数，得到n个范数，此时 `norm` 是一维的， `norm[:,np.newaxis]` 扩展维度到n行1列，就可以直接做除法了。

4. `def draw(self)`

- 绘制特征点
- 调用 `cv2.drawKeypoints`

（2）数据集划分

load_data（）函数

1. 分别读取每个类别的文件名列表

- `for root,dir,files in os.walk(config.image_dir)`

2. 文件名称按照"1.jpg""2.jpg"排序，原读取顺序为"1.jpg""10.jpg"

- `files=sorted(files,key=lambda x:int(x.split('.')[0])`

3. 取前150张图片为训练集，其余为测试集
4. 计算训练集和测试集图片的特征点向量，将全部信息组成两个列表，列表的元素为ImageProcess类的对象

- `return train_image,test_image    #[ImageProcess]`

（3）生成词袋

Bow类

1. `class Bow()` :将n类特征点的中心点作为视觉词汇，生成词袋

Attributes

- numOfBag: 词袋大小
- feature_set: 特征点集合

2. `def generateBoW(self) -> np.ndarray`

- 生成词袋，并保存到默认路径
- 返回值 `centers` ，得到的是 `numOfBag` 行128列的矩阵，它是从 `feature_set` 中的众多特征点向量中，聚类得出的 `numOfBag` 个视觉词汇。

3. `def getBow(self, path):`

- 读取 `path` 路径的词袋文件

（4）SPM算法生成图片的特征表示，构建特征向量数据集

SPM类

1. `class SPM()` :使用SPM算法,生产图片的特征向量,构建数据集

Attributes

- image_set: 图片信息的集合。
- centers: 聚类中心也就是词汇表。

2. `def __init__(self,image_set,centers) -> None`

- 初始化

3. `def single_compute_histogram(self, features, keypoints, image)`

- 对于每张图片，统计不同尺度下图像的特征直方图,并合并。
- 先计算4*4的尺度，给每个分块的每个中心点投票。
- 2*2和1*1都可以由4*4做和而成，不用再另外计算。
- 合并三个尺度的特征时有一个系数。

4. `def generate_dataset(self)`

- 生成图片特征向量数据集。
- 读取整个数据集的特征点，分别调用 `single_compute_histogram` 方法。

（二）图片分类

(1) 支持向量机算法

model Python

```python
def train(dataset,labels):
    print("---模型训练---")
    svc=SVC(kernel='linear',C=1000,decision_function_shape='ovo')
    svc.fit(dataset,labels)
    return svc
```

(2) 预测和评估结果

evaluate Python

```python
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import matplotlib.pyplot as plt
def train(dataset,labels,config):
    print("---模型训练---")
    svc=SVC(kernel=config.kernel,C=config.C,decision_function_shape='ovo')
    svc.fit(dataset,labels)
    return svc
def single_predict(data,model):
    """预测单独一张图片的结果
    """
    data=data.reshape(1,-1)
    prediction = model.predict(data)
    return prediction
def predict(dataset,model):
    """预测数据集的结果
    """
    prediction = model.predict(dataset)
    return prediction
def evaluate(true_labels, prediction, target_names):
    '''评估模型预测结果,生成分类报告和混淆矩阵
    '''
    report = metrics.classification_report(true_labels, prediction, target_names=target_names)
    confuse_matrix = confusion_matrix(true_labels, prediction)
    return report, confuse_matrix
def plot_matrix(cm,classes,num):
    """可视化混淆矩阵
    """
    #转成浮点数
    cm=cm.astype(np.float32)
    #计算每个真实类别的样本数
    s=np.sum(cm,axis=1).astype(np.float32)  #[15,]
    s=s.reshape(-1,1)
    cm=cm/s*100
    fig=plt.figure()
    sub = fig.add_subplot(111)
    color = sub.matshow(cm)
    fig.colorbar(color)
    for i in range(len(cm)):
        sub.text(i,i,format(cm[i][i],'.2f'),va='center',ha='center',fontsize=6)
    sub.set_xticks(range(len(classes)))
    sub.set_yticks(range(len(classes)))
    sub.set_xticklabels(classes, rotation=90)
    sub.set_yticklabels(classes)
    sub.set_xlabel('Predicted labels')
    sub.set_ylabel('True labels')
    plt.savefig(f'cm_{num}.png')
    plt.show()

```

三、环节和参数分析

（一）环节分析

1. 提取sift特征点的过程是固定的。
2. 生成词袋时，通过改变词袋的大小，影响视觉词汇的个数。
3. SPM算法生成特征向量的维度也和词袋大小有关。
4. 训练支持向量机时，需要选择一系列参数

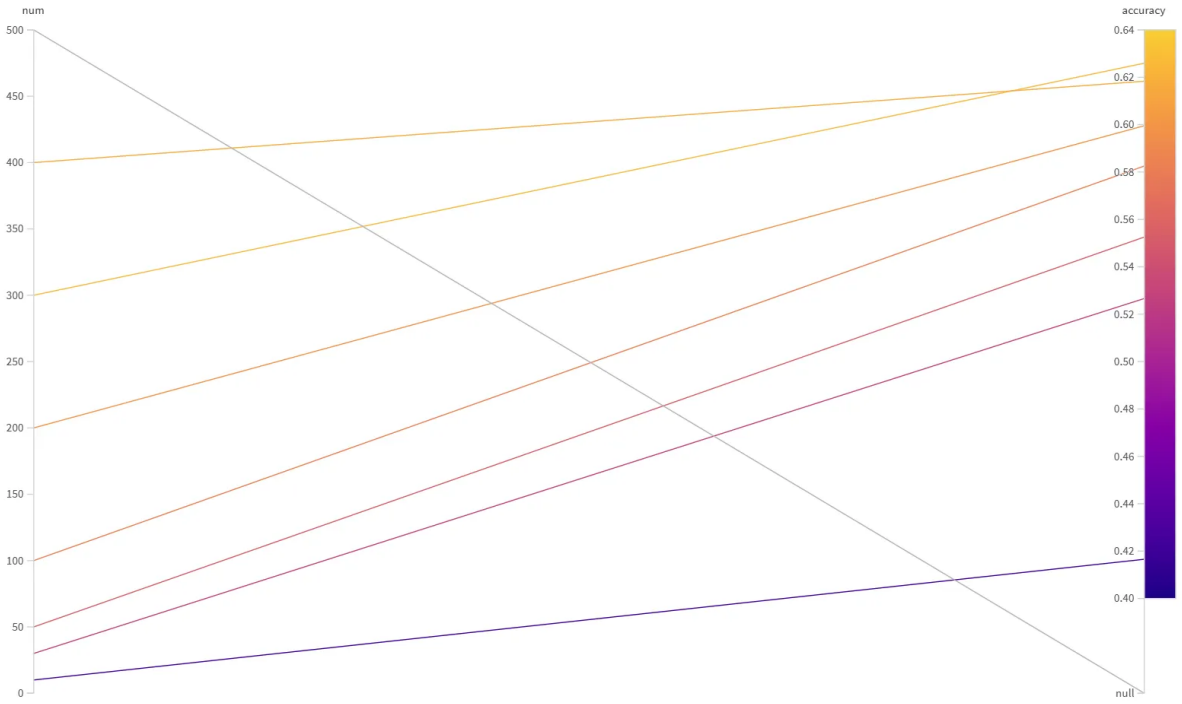
- 正则化参数C
- gamma
- kernel

（二）参数分析

借助 `wandb` 库的可视化工具进行超参数搜索。

（1）词袋大小

随着词袋慢慢增大，分类的效果有着显著提高，增大到一定程度后，再继续增大，训练时间严重变长，而且性能不再有明显优化。当 `vocab_size=500` 时，很长时间都结束不了。之后都选择词袋大小为300，效果比较好。



- **增加视觉词汇量：**增加词袋的大小可以**引入更多的特征**，从而提供更多的信息来训练模型。
- **维度灾难：**随着词袋大小的增加，特征空间的维度也会增加。这可能导致模型变得更加复杂，训练时间变长，并且增加了过拟合的风险。

（2）kernel

1. linear

选择线性核的原因：1)图片的特征向量中大部分元素为零，是稀疏向量。**2)**不用调参数。
可以看到取一张图片的特征向量，4200的维度上，有3647个值为0的元素。

```
1  x=train_set[0]
2  print(len(x))
3  print(list(x).count(0))
✓  0.0s

4200
3647
```

- 线性核的 SVM 在处理稀疏向量时通常效果很好
- 线性核不需要进行昂贵的特征映射或距离计算，因此在处理稀疏向量时的计算复杂度较低，训练时间短一些。

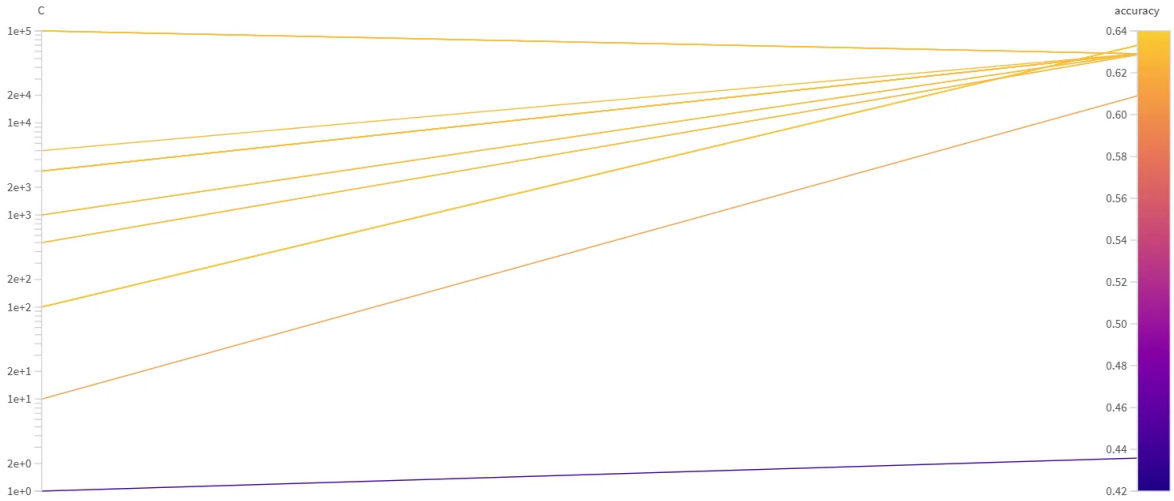
2. rbf

数据量不是特别大，所以也可以选择rbf核。

- RBF 核可以通过将数据映射到高维空间来更好地捕捉特征之间的非线性关系。
- RBF 核对应的决策边界可以更灵活地适应数据的分布

(3) C

正则化参数 C 在支持向量机中是用来控制模型的复杂度和拟合程度的重要参数。C 的大小直接影响了模型在训练过程中对误分类样本的惩罚程度。
gamma取默认值
`C: "values":[1,10,100,500,100,3000,5000,10000,100000]`
随着C值增大，准确率在慢慢提升。



1. **小的 C 值**：

- 较小的 C 值意味着对误分类的惩罚较小，模型更倾向于选择较大的间隔来容忍一些误分类样本
- 模型更倾向于选择简单的决策边界，减少了对训练数据的过度拟合。

2. **大的 C 值**：

- 较大的 C 值意味着对误分类的惩罚较大，模型更倾向于尽可能地正确分类每个训练样本，这可能会导致准确率更高，但也会**增加训练时间**。
- 模型可能会过于关注于最大化间隔而忽略了一些训练样本的分类情况，导致了欠拟合的问题。

（4）gamma

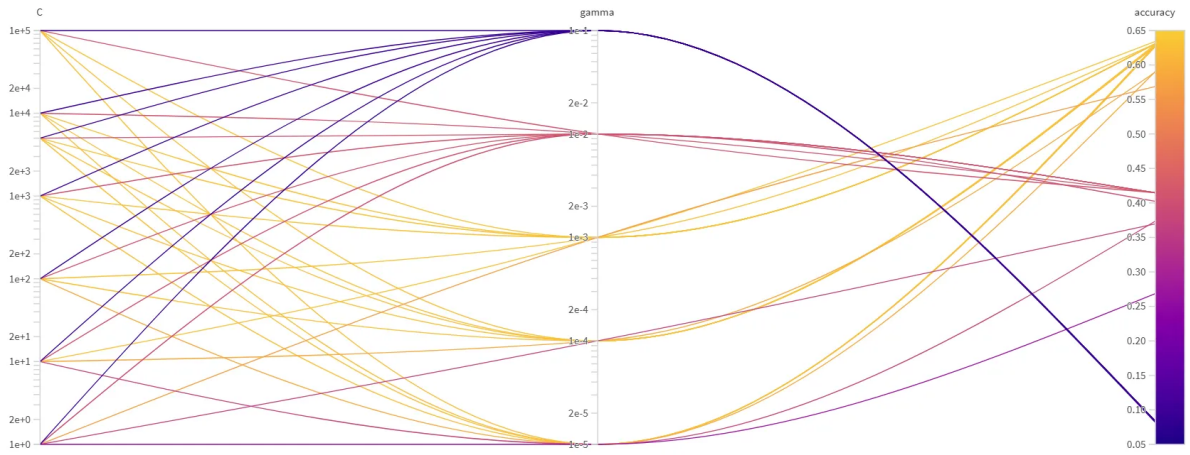
gamma 参数是 RBF 核的一个重要参数，它控制了样本在高维特征空间中的聚集程度。一般来说，较高的 gamma 值通常被认为是大于默认值（通常是 1 / 特征数量）的值，在这个任务中 $\frac{1}{2250}$ 数量级在1e-3。

1. **小的 gamma 值**：
- 较小的 gamma 值意味着样本在高维特征空间中的聚集程度较低，决策边界相对平滑，会导致模型更简单。

2. **大的 gamma 值**：
- 较大的 Gamma 值会导致模型更加复杂，因为样本点之间的相似度更高，决策边界更为复杂。
- 在训练样本较少的情况下，选择过大的 Gamma 值可能会导致模型过度拟合，降低了模型的泛化能力。

```
sweep_config = {
"method": "grid",
"metric": {"name": "accuracy", "goal": "maximize"},
"parameters": {
    "gamma":{"values":[0.00001,0.0001,0.001,0.01,0.1]},
    "C":{"values":[1,10,100,1000,5000,10000,100000]}
}
}
```

（5）C和gamma关系



- 较为平滑的模型（gamma 较小）可以通过增加 C 值来使得模型变得复杂一些，去拟合多一些的样本。（比如图中的**1e-5**）。
- 注意到，对于某些较为适中的 gamma 值（比如图中的**1e-3**）。当 C 变得非常大的时候，也能得到跟 C 比较小的时候一样表现好的模型。
- 但是， RBF 核辐射范围即 gamma 比较合适时，核本身就可以作为一个很好的结构性正则途径。所以如果当 C 取得比较小的时候模型的表现就已经不错的前提下，不倾向于把 C 取得非常大，因为 C 值较小能够节省内存，而且跑的更快。

四、实验结果展示

根据以上分析得到两个较好的模型

`svc=SVC(kernel=config.kernel, C=config.C, gamma=config.gamma, decision_function_shape='ovo')`

词袋大小均选择 `300`

1. linear

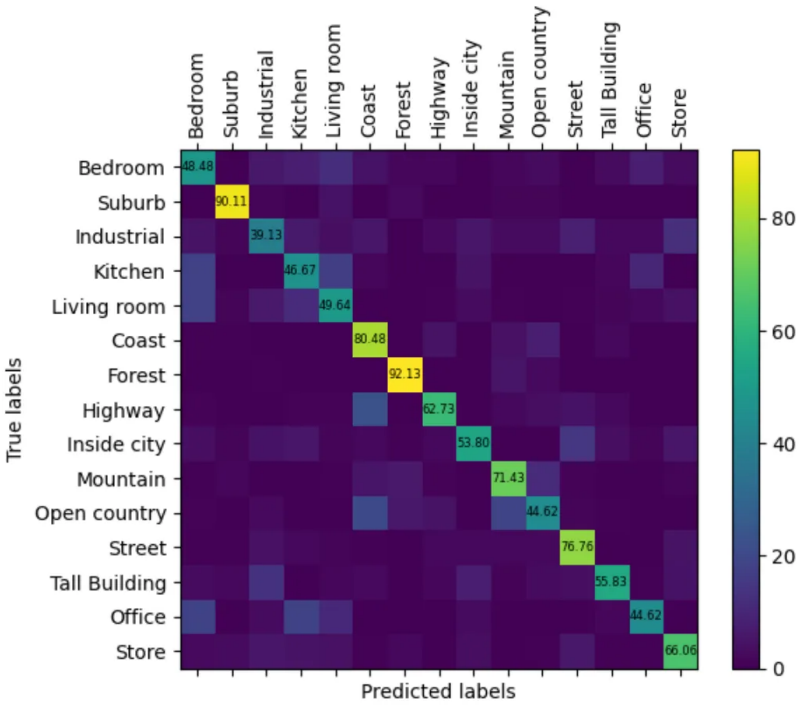
训练的速度会比 rbf 核快。

2. rbf

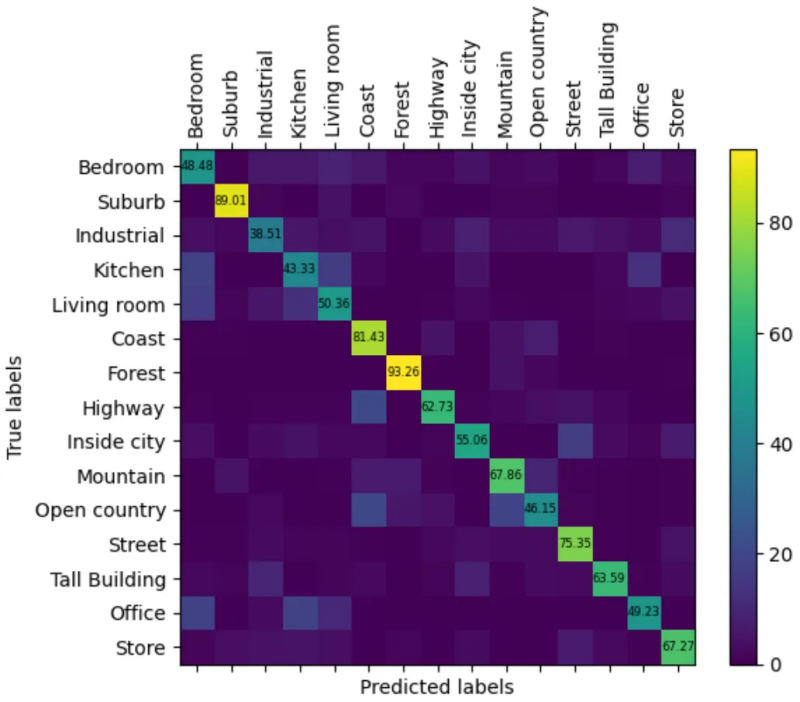
Agent	State	Notes	Use	Tag	Create	Runtim	Sweep	C	gamma	kernel	num	acc
k71wiqz6	Finished	Add notes	whyb1j		16h ago	1m 24s	c-gamma	10	0.001	-	-	0.634

（一）最终的混淆矩阵

1. linear



2. rbf



（二）分类结果

1. linear

测试集混淆矩阵：

	precision	recall	f1-score	support
Bedroom	0.29	0.48	0.36	66
Suburb	0.80	0.90	0.85	91
Industrial	0.46	0.39	0.42	161
Kitchen	0.30	0.47	0.37	60
Living room	0.57	0.50	0.53	139
Coast	0.60	0.80	0.69	210
Forest	0.82	0.92	0.87	178
Highway	0.63	0.63	0.63	110
Inside city	0.66	0.54	0.59	158
Mountain	0.66	0.71	0.69	224
Open country	0.63	0.45	0.52	260
Street	0.63	0.77	0.69	142
Tall Building	0.83	0.56	0.67	206
Office	0.58	0.45	0.50	65
Store	0.66	0.66	0.66	165
accuracy			0.63	2235
macro avg	0.61	0.61	0.60	2235
weighted avg	0.64	0.63	0.62	2235

2. rbf

测试集混淆矩阵:

	precision	recall	f1-score	support
Bedroom	0.32	0.48	0.39	66
Suburb	0.75	0.89	0.81	91
Industrial	0.51	0.39	0.44	161
Kitchen	0.30	0.43	0.35	60
Living room	0.59	0.50	0.54	139
Coast	0.60	0.81	0.69	210
Forest	0.82	0.93	0.87	178
Highway	0.64	0.63	0.64	110
Inside city	0.64	0.55	0.59	158
Mountain	0.65	0.68	0.66	224
Open country	0.66	0.46	0.54	260
Street	0.62	0.75	0.68	142
Tall Building	0.83	0.64	0.72	206
Office	0.57	0.49	0.53	65
Store	0.67	0.67	0.67	165
accuracy			0.63	2235
macro avg	0.61	0.62	0.61	2235
weighted avg	0.64	0.63	0.63	2235