实验报告：图像分类实验

1. 引言

图像分类是计算机视觉领域的基础任务之一，其目标是将图像分配到预定义的类别中。在本实验中，我们探索了一种基于SIFT 特征提取、K聚类、词袋表示、支撑向量机的图像分类方法。

2. 实验目标

本实验的主要目标是：

- 实现基于SIFT特征提取和SVM分类器的图像分类器。
- 对图像数据进行预处理、特征提取、特征聚类和模型训练等步骤。
- 保存训练好的模型参数，以便于随时加载模型。

3. 实验方法

3.1 数据准备

我们使用了一个名为`scene_categories`的数据集，`scene_categories`数据集常用于图像分类、目标检测和其他计算机视觉任务的研究和评估。该数据集由15个场景类别的图像组成，每个类别包含约200张图像。这些图像是在真实世界中拍摄的，涵盖了各种不同的场景，包括城市街道、沙滩、山脉、森林等。

`scene_categories`数据集通常用于评估图像分类算法的性能，研究者可以利用该数据集进行模型训练和测试，并根据分类准确率等指标评估算法的性能。由于该数据集具有真实世界场景的图像，因此能够更好地反映模型在实际场景中的应用效果。

我们用每个类中编号前 150 号的样本作为训练样本，15 个类一共 2250 张训练样本；剩下的样本构成测试集合。

```
# 1. 数据准备
......
for category in os.listdir(train_path):
    for filename in os.listdir(os.path.join(train_path, category))[:150]:
        ......
        train_images.append(img_gray)
        train_labels.append(category)
        ......
```

这段代码通过遍历训练集文件夹中的每个类别和其中的图像文件，将前150张图像作为训练集，后续图像作为测试集，将每张图像的灰度数据存储在对应的训练或测试图像列表中，同时将图像所属的类别作为标签存储在对应的训练或测试标签列表中，以便后续的模型训练和评估。

3.2 特征提取和表示

我们采用了SIFT特征提取方法，并使用空间金字塔对图像进行多尺度特征提取。

```
# 2. 特征提取和表示（使用空间金字塔）
def extract_features_with_pyramid(images):
    sift = cv2.SIFT_create()
    ......
            resized_img = cv2.resize(img, (img.shape[1] // scale, img.shape[0] //
scale))
            kp, des = sift.detectAndCompute(resized_img, None)
            ......
```

这段代码定义了一个名为 `extract_features_with_pyramid()` 的函数，用于对输入的图像列表进行特征提取和表示，采用了空间金字塔的方法。函数内部循环遍历每张图像，构建三级空间金字塔，对每个金字塔级别的图像利用 SIFT 特征提取器提取关键点并计算特征描述子，最终将所有图像的特征描述子存储在一个列表中并返回。在主程序中，分别对训练集和测试集的图像调用该函数，以准备后续的特征聚类和分类器训练。

`cv2.resize(img, (img.shape[1] // scale, img.shape[0] // scale))` 这行代码通过OpenCV中的`cv2.resize()`函数对图像进行缩放操作，将原始图像`img`按照指定的缩放因子`scale`进行缩放，并将缩放后的图像存储在`resized_img`中，以构建空间金字塔中不同级别的图像。

构建空间金字塔的优势在于它可以提高图像处理算法对不同尺度图像的适应性，使得算法在检测和识别不同尺度下的目标或特征时更加准确和鲁棒，从而提升了图像处理任务的性能。然而，构建空间金字塔也存在一些不足之处，比如增加了计算复杂度和存储需求。

`sift.detectAndCompute()` 是SIFT特征提取器的一个函数，用于同时检测图像中的关键点（Key Points）并计算这些关键点的特征描述子（Descriptors）。函数执行后，返回两个值 `kp` 和 `des`，分别表示检测到的关键点和对应的特征描述子。`kp` 是一个列表，每个元素是一个关键点对象，包含关键点的位置、尺度等信息；`des` 是一个二维数组，每一行是一个关键点的特征描述子向量，用于描述关键点周围的局部特征信息。SIFT算法默认生成的特征描述子的维度是128维。

3.3 特征聚类

我们使用K均值聚类对提取的特征进行聚类，得到视觉词汇（Visual Words）。

```
# 3. 特征聚类
all_descriptors = [desc for sublist in train_descriptors for desc in sublist]
kmeans = KMeans(n_clusters=500)
kmeans.fit(all_descriptors)
```

这段代码将训练集图像的特征描述子通过K均值聚类方法聚类为500个类别，生成了一组视觉词汇（Visual Words），用于后续的图像表示和分类。

`KMeans(n_clusters=500)` 初始化了一个K均值聚类器对象，其中 `n_clusters=500` 表示要将特征描述子聚类成500个类别。K均值聚类是一种无监督学习算法，通过迭代将数据点分配到K个簇中，使得每个数据点与所属簇的中心点（即聚类中心）的距离最小化。在这里，500代表了聚类的簇数，即聚类后将得到500个聚类中心，也就是500个视觉词汇，用于表示图像的特征。

增加视觉词汇的数量有助于提高特征的丰富性，但并非总是理想的选择。这取决于具体的应用场景和算法设计，因为增加视觉词汇会增加计算复杂度、存储需求，并可能导致维度灾难、过拟合和聚类质量下降等问题，因此在选择视觉词汇数量时需要进行权衡和调优。

3.4 构建词袋表示

这一步，我们将图像表示为一个固定长度的特征向量，其中每个元素对应于聚类中心的计数。

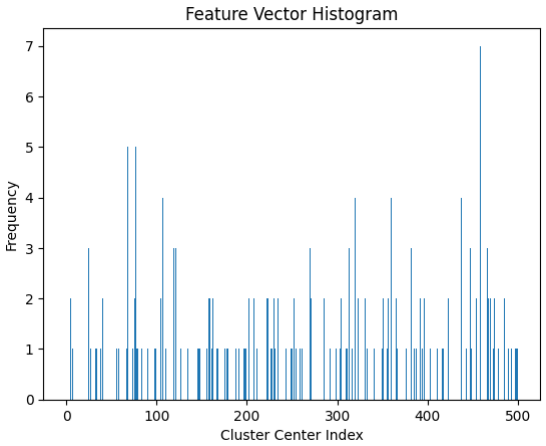
```
def build_features(descriptors, kmeans):
    ......
            labels = kmeans.predict(desc)
            for label in labels:
                histogram[label] += 1
        features.append(histogram)
    ......
```

这段代码定义了一个名为 `build_features` 的函数，用于构建图像的特征向量。函数接受两个参数：

1. `descriptors`：图像的特征描述子列表，每个元素是一个包含特征描述子的数组。
2. `kmeans`：K均值聚类器对象，用于对特征描述子进行聚类。

这个函数 `build_features` 接受图像的特征描述子列表和K均值聚类器对象作为输入，然后通过遍历每个特征描述子，利用K均值聚类器对其进行聚类，得到聚类标签后构建特征描述子的直方图表示，最终返回一个包含所有训练或测试图像的特征向量的列表。

函数 `plot_histogram(histogram)` 的作用是绘制特征向量的直方图。函数接受一个特征向量 `histogram` 作为输入。下面是一个实例，词袋大小是500：



3.5 训练分类器

我们使用支持向量机（SVM）作为分类器，并将得到的词袋表示作为输入特征进行训练。

```
svm = SVC()
svm.fit(train_features, train_labels)
```

这两行代码创建了一个支持向量机（SVM）分类器对象 `svm`，并使用训练集的特征向量 `train_features` 和标签 `train_labels` 对其进行训练，以学习如何将特征向量映射到相应的类别，从而构建了一个用于图像分类的分类模型。

然后我们可以使用下面的命令保存模型参数：

```
joblib.dump(svm, 'pkl/svm_model.pkl')
joblib.dump(kmeans, 'pkl/kmeans_model.pkl')
```

这两行代码训练好的支持向量机分类器对象 `svm` 和 K均值聚类器对象 `kmeans` 分别保存到名为 `svm_model.pkl` 和 `kmeans_model.pkl` 的文件中，以便在以后的应用中可以重新加载模型并使用。

3.6 评估分类器

我们对训练好的分类器进行测试，并计算分类准确率作为评估指标。

```
predictions = svm.predict(test_features)
accuracy = accuracy_score(test_labels, predictions)
```

这两行代码首先利用训练好的支持向量机模型 `svm_model` 对测试集的特征向量 `test_features` 进行预测，得到对应的预测结果 `predictions`。然后，利用预测结果 `predictions` 和测试集的真实标签 `test_labels` 计算分类准确率 `accuracy`，以评估模型在测试集上的分类性能。

```
# 计算混淆矩阵
conf_matrix = confusion_matrix(test_labels, predictions)
```

这行代码用于计算混淆矩阵，即评估分类器性能的一种方式，通过比较模型预测的类别和真实类别的对应关系来衡量分类器的准确性。`confusion_matrix` 函数接受真实标签 `test_labels` 和模型预测结果 `predictions` 作为输入，然后返回一个矩阵，其中行表示真实类别，列表示预测类别，矩阵中的每个元素表示对应类别样本的数量。

4. 实验结果

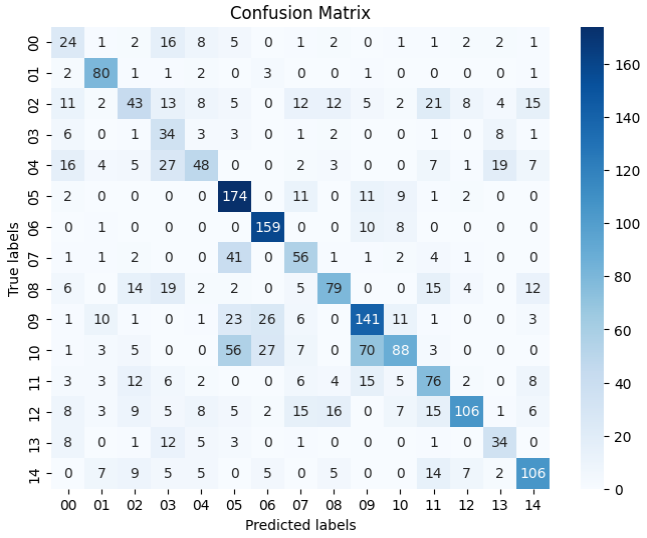
实验结果表明，我们的图像分类器最高在测试集上达到了55.83%的准确率。

表格中，K表示聚类数，L表示空间金字塔层数

参数	k=50，L=1	k=100,L=1	k=100,L=2	k=100,L=3	k=500,L=1
accuracy	0.5248322147651007	0.5360178970917227	0.5516778523489932	0.548993288590604	0.5583892617449664

本次实验由于运算速度有限，没有测试更多的参数。

下面是k=500,L=1得到的混淆矩阵：



5. 结论与讨论

本实验中，我们成功实现了基于SIFT特征提取和SVM分类器的图像分类方法。通过实验结果的分析，我们发现：

1. **聚类数量对性能的影响：** 在相同的空间金字塔层数下，随着聚类数量从50增加到500，准确率有所提高。这表明增加聚类数量有助于提高特征表示的表达能力，使得模型能够更好地捕捉图像的视觉特征，从而提高分类性能。
2. **空间金字塔层数对性能的影响：** 在相同的聚类数量下，随着空间金字塔层数从1增加到3，准确率略有波动，但整体变化不大。这可能是因为在该数据集上，增加空间金字塔的层数未能有效提高特征的表达能力，或者增加的层数引入了过多的冗余信息，影响了模型的性能。

各个环节和参数对最终性能的影响：

1. **特征提取方法：** 使用SIFT特征提取方法是一个重要的环节，SIFT特征的选择直接影响到后续特征表示的质量和分类器的性能，可能需要尝试其他特征提取方法。
2. **聚类数量：** K均值聚类中的簇数是一个重要的参数，选择不同的簇数会影响到词袋模型的表示能力和区分度。过多的簇数可能导致视觉词汇过多、特征维度过高，增加模型复杂度和计算开销；而过少的簇数可能无法充分捕获图像的多样性特征，导致信息丢失和分类性能下降。
3. **特征表示方法：** 词袋模型是一种简化的特征表示方法，忽略了单词之间的空间信息和顺序，可能无法充分表达图像的结构和局部特征。
4. **分类器选择和参数调优：** SVM作为线性分类器在某些情况下可能表现不佳，特别是在非线性可分问题上。选择合适的分类器，并对其参数进行调优，如核函数的选择、正则化参数的设置等，可以显著影响分类器的性能。

6. 实验总结

本实验对图像分类任务的探索提供了一种基于传统特征提取和机器学习方法的解决方案。然而，我们也发现这种方法似乎距离准确分类还有一段距离。这种基于传统特征提取和机器学习的图像分类方法，在复杂场景下可能准确度不高。

一些可能的改进的方向或更好的算法包括：

1. **深度学习方法：** 使用深度学习模型，如卷积神经网络、循环神经网络等，可以自动学习图像中的高级特征表示，避免了手动设计特征提取器的复杂性，并在许多图像分类任务中取得了显著的性能提升。
2. **迁移学习：** 基于预训练的深度学习模型进行迁移学习，将已经在大规模数据集上训练好的模型权重迁移到目标任务中，可以加速模型训练过程，并提高模型在小规模数据集上的性能。
3. **集成学习：** 使用集成学习方法，如随机森林、梯度提升树等，将多个基础分类器的预测结果进行组合，可以提高分类器的稳定性和泛化能力。
4. **特征学习：** 结合传统特征提取方法和深度学习模型进行特征学习，可以在保留传统特征的解释性的同时，利用深度学习模型学习更高级别的特征表示。
5. **自监督学习：** 利用自监督学习方法，如对比学习、生成式对抗网络（GAN）等，可以在无监督的情况下学习有用的特征表示，从而提高图像分类的性能。