Experiment Report on Scene Image Classification Based on the Bag-of-Words Model

1 Overview

This experiment aims to explore the performance of combining Scale-Invariant Feature Transform (SIFT) and Bag-of-Words Model (BoW) with Support Vector Machine (SVM) on the classification task of the 15-Scene dataset. Through hyperparameter search, we determined the optimal parameter combination, further improving the model's classification accuracy.

2 Experimental Data and Process Overview

2.1 Dataset Image Overview

The experiment uses a dataset of 15 different scenes, including natural landscapes and indoor scenes.

图 1　Dataset Image Overview

The first 150 images of each category are used as the training set, with the remaining images used for validation to ensure the fairness and reliability of the experimental results.

2.2 Algorithm Process

The scene image classification algorithm process includes the following key steps:

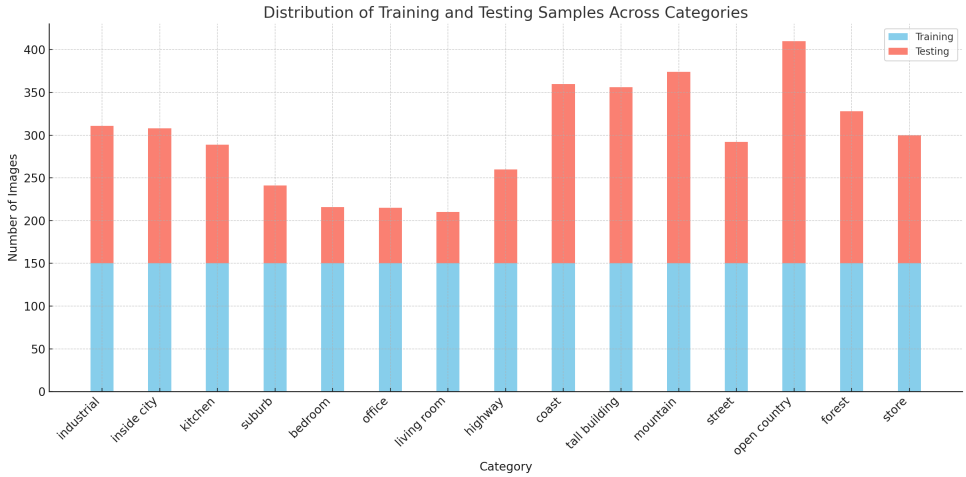


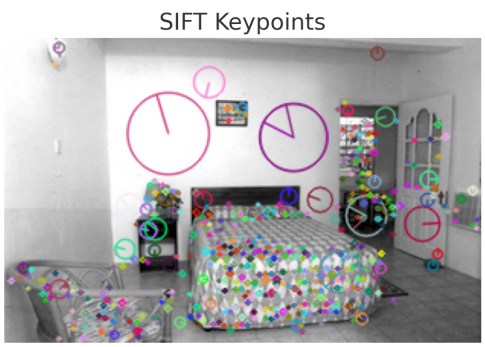
图 2　Dataset Division

1. **Data Loading**: Implement a data loader to split the image dataset into training and test sets, ensuring each category of images is proportionally allocated so the model can learn the features of each scene.
2. **Feature Extraction**: Use the Scale-Invariant Feature Transform (SIFT) algorithm to extract keypoints and descriptors from each image. This step is crucial for capturing local features of the images and is essential for subsequent classification results.
3. **Feature Encoding**: Use the KMeans clustering algorithm to encode the extracted features, generating a visual vocabulary. Each image is encoded into a fixed-length feature vector through these vocabularies. Convert the SIFT descriptors into a histogram representation.
4. **Classifier Training**: Use the Support Vector Machine (SVM) algorithm to learn from the feature-encoded training data to build a model that can classify images into different scenes.
5. **Result Evaluation**: Classify the test set and evaluate the model's performance by calculating the classification accuracy and generating a confusion matrix.

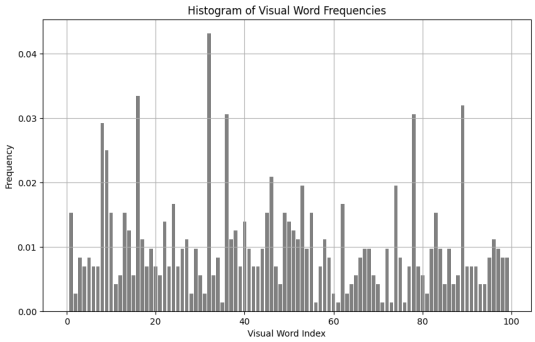
3 Functionality and Parameter Explanation

- `fetch_dataset(dataset_dir, download_url)`
  Function: Download and unzip the dataset, ensuring data integrity.
- `load_data(path, split, nOctaveLayers)`
  Function: Load data from the directory and split it into training and test sets according to the given ratio.
  Parameter Explanation: The ttnOctaveLayers parameter significantly affects feature extraction, determining the number of octave layers in SIFT feature extraction.
- `feature_histogram(descriptors, model)`

Function: Convert descriptors into normalized histogram features using a clustering model.



((a)) SIFT Features



((b)) BoW Statistical Histogram

4 Experimental Results Analysis

4.1 Confusion Matrix Analysis

The confusion matrix reveals the classifier's performance on specific categories. For example, the classifier has a high recognition accuracy on natural scenes such as "coast" and "forest", while it performs relatively poorly on some indoor scenes such as "bedroom" and "living room". This may be due to the more obvious features in natural scenes, whereas indoor scenes have lower distinguishability between different categories.

图 4 Example of Confusion Matrix

5 Model Selection and Hyperparameter Optimization

5.1 SIFT Parameters

In this experiment, adjusting the `nOctaveLayers` parameter had a significant impact on classification performance. The experimental results showed that the best classification effect was achieved when this parameter was set to 5. A larger `nOctaveLayers` value means that more layers are generated in each octave, allowing the SIFT feature extraction to capture more levels of image detail and texture information, thereby improving the detection and description of keypoints, especially in texture-rich scenes.

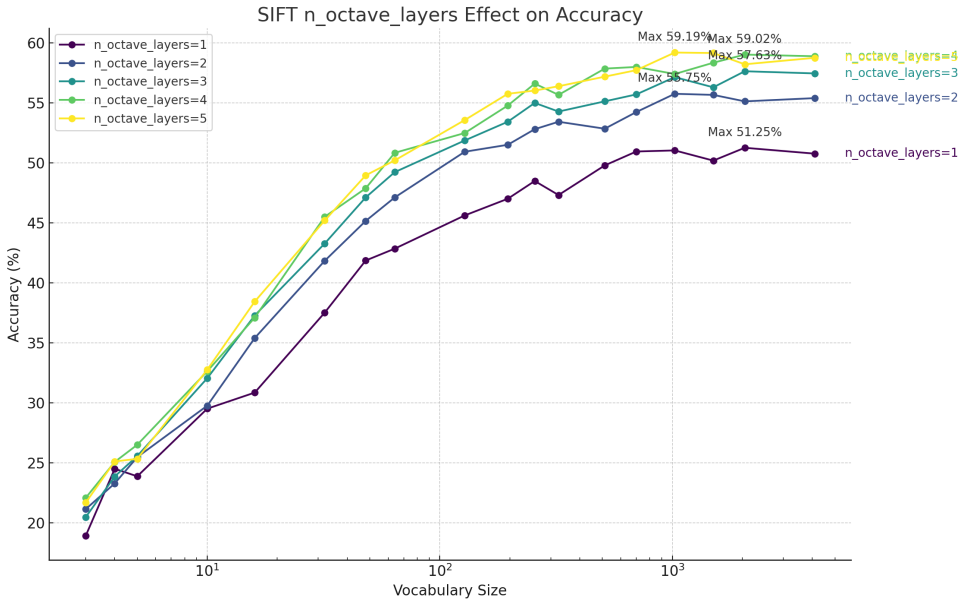


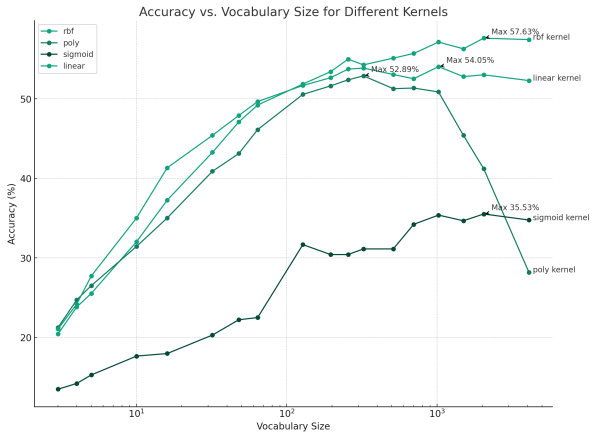
图 5　Classification Performance Curve under Different `n_octave_layers` Settings

5.2 Kernel Function Selection and Cluster Size

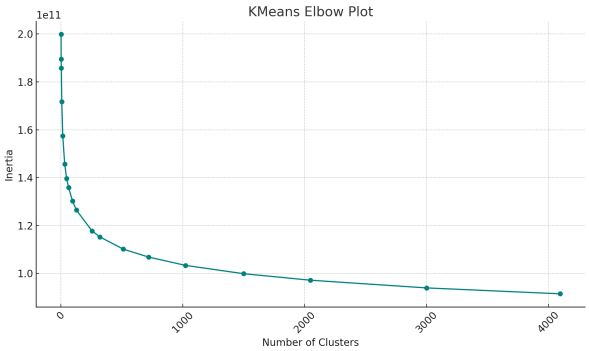
In the Support Vector Machine (SVM) classifier, the radial basis function (RBF) kernel, due to its provision of nonlinear mapping, performs excellently in complex scene classification tasks, outperforming other kernels. The vocabulary size —the number of clustering clusters — directly affects the diversity and discrimination ability of feature encoding.

5.3 Hyperparameter Search

Through 200 hyperparameter searches, we determined the best parameter combination. The search space included kernel function type, vocabulary size, regularization parameter C, etc. The final optimal parameter combination was: kernel function as RBF (`kernel='rbf'`), vocabulary size of 2580 (`vocabulary_size=2580`), regularization parameter of 10.0 (`C=10.0`), fea-



((a)) Performance Comparison under Different Kernel Functions and Vocabulary Sizes



((b)) Elbow Method Graph for KMeans Clustering Cluster Size Selection

图 6 Performance Comparison of Different Clustering and Parameter Optimization Methods

ture extraction method as histogram (`func='histogram'`), SIFT octave layers set to 5 (`nOctaveLayers=5`), and batch size of 500 (`batchsize=500`). Under this parameter setting, the model achieved a highest accuracy of 60.72

6 Improvement Measures and Future Directions

This section outlines possible improvement measures and future research directions aimed at enhancing the performance and efficiency of the image classification system.

- **MiniBatchKMeans**: Used to accelerate the clustering process
  . By using small batches of data, the computation time and resource consumption can be significantly reduced.
- **Caching Mechanism**: Stores feature extraction and clustering results, avoiding redundant computations. This can improve the overall efficiency of the system, especially when dealing with large datasets.
- **Pooling Methods**: Average pooling and max pooling provide a simplified feature representation method, achieving a similar effect to BoW. The following chart shows the performance comparison of the two pooling methods under different settings. The use of average pooling performed better than max pooling, possibly because max pooling is susceptible to image noise, as noise points may become the maximum values within a region. In contrast, average pooling is less sensitive to noise, as noise is diluted when averaging.
- **Attention Mechanism**: Explore attention-based feature aggregation methods to obtain a global feature representation. This approach is expected to improve the model's sensitivity to key information, thereby enhancing classification performance.

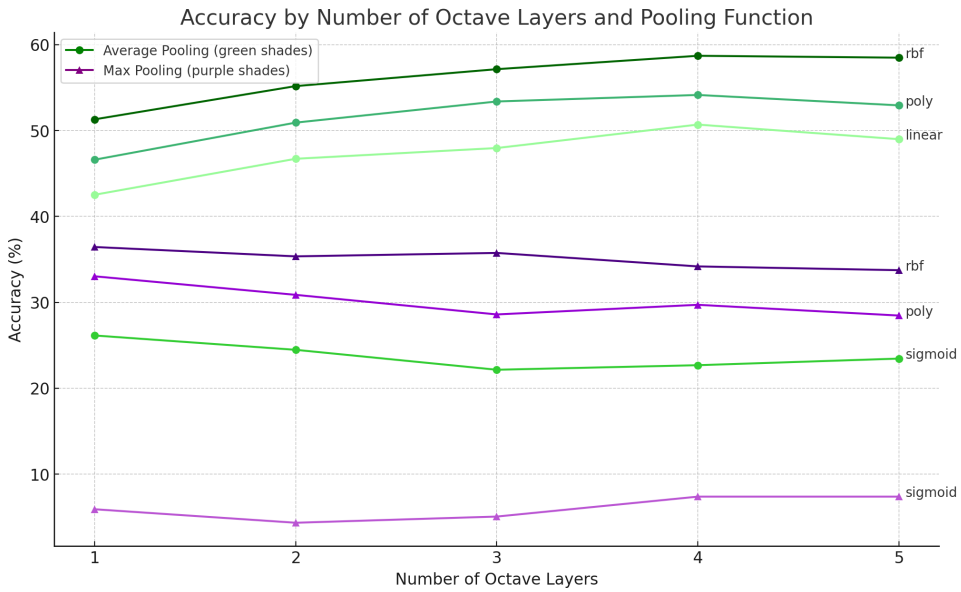


图 7　Performance Comparison of Pooling Methods

- **Feature Extractor Exploration**: Try other feature extractors such as ORB and AKAZE, although their performance did not surpass SIFT. Further research into these new methods may reveal their advantages in specific applications.
- **Multi-Model Fusion**: Use ensemble learning methods, such as model stacking and model voting, to combine the strengths of multiple classification models. This approach can generally enhance the stability and accuracy of the classification system by integrating different perspectives and interpretations of the data by different models, reducing potential biases and overfitting.

A Appendix

1.1 Code File Explanation

The following table lists the main code files used in this project and their function descriptions:

File Name	Description
main.py	Main implementation, including Scale-Invariant Feature Transform (SIFT), Bag-of-Words Model (BoW), Support Vector M
plot.py	Responsible for generating various charts for experimental results.
params_search.py	Used for hyperparameter search to determine the optimal model configuration.

表 1　Project Code File Explanation

1.2 Performance Comparison with Other Methods

Our implementation of BoW on the 15-Scene dataset achieved a classification accuracy of 60.72

Algorithm	**Classification Accuracy (Dictionary Size)**
BoW [6]	65.87 ± 0.61 (90)
ScSPM [38]	79.54 ± 0.70 (420)
LLC [36]	80.85 ± 1.02 (420)
VLAD [17]	77.35 ± 0.50 (400)
VLAD+BoW	80.09 ± 0.51 (280)
MVLAD	78.82 ± 0.50 (280)
TNNVLAD	79.23 ± 0.62 (400)
Our BoW Implementation	*60.72%*

表 2　Classification Accuracy Comparison on 15-Scene Dataset