Universiteit Van Amsterdam

Final Lab (Part 1)

Keypoint Detection, Bag of Visual Words and Image Classification

Authors: Robert-Ștefan Sofroni (16300645)
Ippokratis Pantelidis (16124006)
Andor Károly Bodgál (14339617)

Group Number: 56
Course: Computer Vision 1

October 27, 2025

Section 1: Data Preparation

The goal of the project was to implement an image classification system that could identify objects given a set of classes. For this implementation, we focused on a **5-class** image classification using **Bag-Of-Visual-Words** approach.

Classes

The group was given a total of 5 classes: **Frog, Automobile, Bird, Cat, and Deer**.

Dataset

CIFAR-10 was the dataset provided to the group for this project. It contains a total of 10 classes and a combination of 60,000 images (32x32), where 50,000 were used for training and 10,000 for testing.

Section 2: Keypoint Detection and Feature Extraction

The primary goal of this section of the project was to establish the foundation of the BOVW. We searched for distinctive keypoints and feature descriptors that characterize local appearance. We started by rescaling the images from 32x32 to 256x256 to make them more meaningful at higher scales. The images were converted to RGB and then to grayscale for SIFT and ORB feature extractors to work on the intensity gradients. For each image, we identified the **Top-N** keypoints, followed by their descriptors.

Methods for Feature Extraction

SIFT

SIFT is a feature extractor that finds keypoints by using the Difference of Gaussians (DoG). The descriptors are derived from the dominant orientations of the keypoints, giving them a dimension of 128. This method is scale and rotation-invariant.

Their parameters:

- `'nfeatures'` - Limits number of top keypoints detected per image.
- `'contrastThreshold'` - Controls sensitivity to low-contrast regions.

ORB

ORB is a feature extraction method that combines the FAST corner detector for keypoints with BRIEF for generating descriptors. Their parameters:

- `'nfeatures'` - Limits number of top keypoints detected per image.
- `'fastThreshold'` - Controls sensitivity of the FAST detector.

Results

Table 1 shows the differences in the number of keypoints detected. For this representation, we used the following values:

- `'contrastThreshold'`: 0.01, 0.03, 0.05
- `'fastThreshold'`: 5, 10, 20

Method	Threshold Type	Threshold	Frog	Automobile	Bird	Cat	Deer
SIFT	contrastThreshold	0.01	138.5	95.5	83.5	119.0	106.0
SIFT	contrastThreshold	0.03	85.5	64.5	39.0	69.0	70.0
SIFT	contrastThreshold	0.05	49.0	46.0	32.0	38.0	41.0
ORB	fastThreshold	5	150.0	150.0	150.0	150.0	150.0
ORB	fastThreshold	10	120.5	149.0	127.5	125.5	126.5
ORB	fastThreshold	20	22.0	49.0	37.5	31.5	36.5

Table 1: Comparison of SIFT and ORB feature extraction results for varying threshold parameters. Each value represents the average number of detected keypoints across two sampled images per class.

By observing the tables, we noticed how, for both ORB and SIFT, the number of keypoints reduced. This is because of their thresholds. For SIFT, the more low-contrast regions we ignore the more we increase the threshold. For ORB, the threshold controls the sensitivity of the FAST threshold. The higher the threshold, the stronger the corner must be for FAST to detect it.

Example

To better understand the differences between SIFT and ORB keypoint detection, we present visual comparisons of both algorithms applied to sample images from our dataset. Figure 1 shows how SIFT and ORB performed on the same image for each one of the classes.

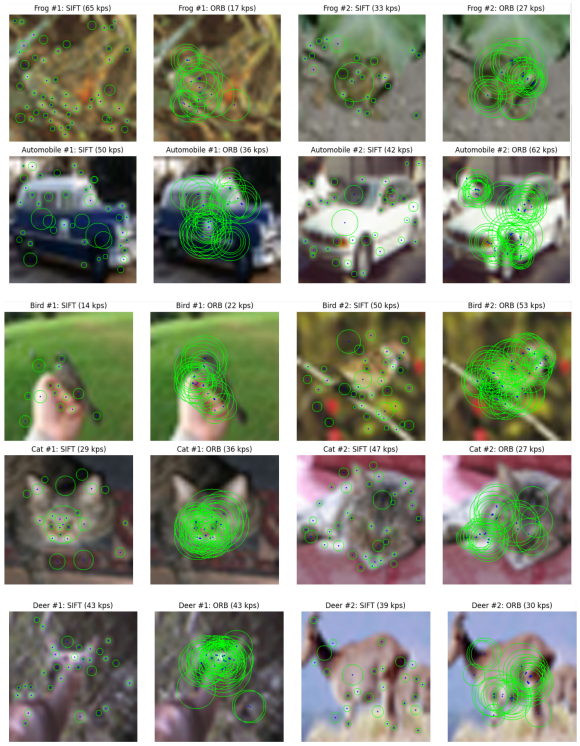


Figure 1: Visual comparison of SIFT vs ORB keypoint detection on different classes.

Section 3: Building the Visual Vocabulary

Objective

The goal of this section is to create the **Visual Vocabulary** for the BOVW by using K-Means. We used the FAISS K-Means clustering algorithm to cluster the descriptors obtained from SIFT and ORB. We had followed the following steps:

1. Use SIFT and ORB on different subsets of the training data (30%, 40%, 50%) to extract the descriptors.
2. Stack all descriptors together.
3. Perform K-Means clustering with FAISS with K = 1000 centroids.
4. Create the visual vocabulary. Each centroid is considered a visual word within the vocabulary.

In order to control the cost of training the K-Means model, we added a `'max_descriptors'` variable that sets a limit for the number of descriptors to **100,000**.

Results

Because FAISS K-Means was given a random seed for initializing the centroid, we will use only a chosen set of parameters for SIFT and ORB:

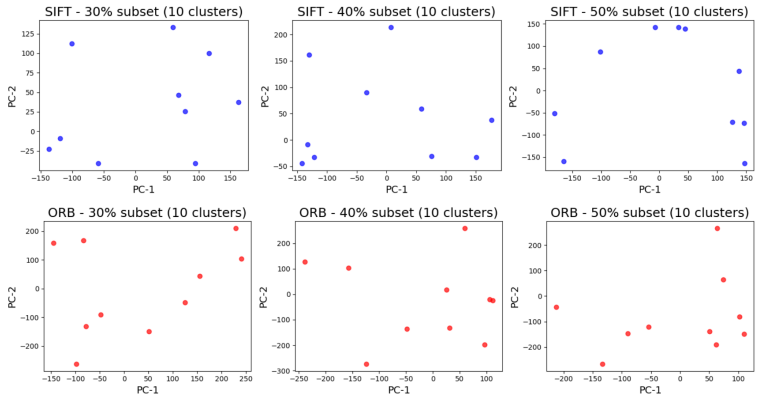


Figure 2: Visualization of the first 10 clusters after K-Means with 20 iterations.

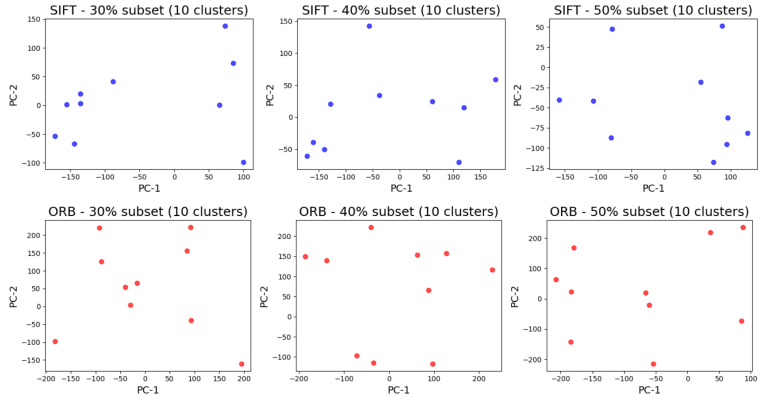


Figure 3: Visualization of the first 10 clusters after K-Means with 50 iterations.

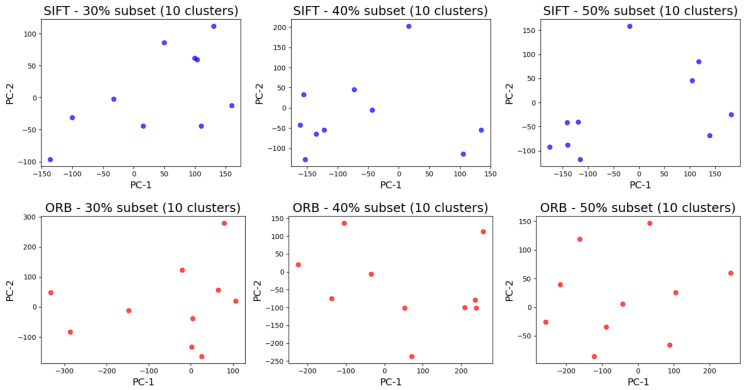


Figure 4: Visualization of the first 10 clusters after K-Means with 100 iterations.

We applied PCA in order to reduce the dimensions of the descriptors to 2D.

However, we can see how the K-Means performs at different numbers of iterations.

Images within Figures 2, 3, 4 reveal the first 10 clusters of each K-means model that was trained with a different number of iterations. A smaller sample of data (e.g., 30%) covers fewer visual patterns, leading to more biased centroids in comparison to the case when using a larger sample. However, increasing the sample of images also results in a larger computational cost. In other words, the more images we use for training, the more generalizable the model will be.

Section 4, 5 and 6: Encoding and Bag of Visual Words

Now that we have constructed the visual vocabulary, the next objective was to represent every training image as a histogram of visual word occurrences. This was done by encoding, a process that transforms features into vectors.

Section 4: Encoding Training Features

For this section, we worked with the vocabulary that was used in 50% of the data. We applied the FAISS nearest-neighbor search to find the clusters that were closest to one another.

Following this step, we created the histograms by counting the number of descriptors for each cluster centroid which now serves as a visual word. This serves as the output of the Bag of Visual Words algorithm. However, because each image comes with a different number of keypoints detected, we applied **L1-Normalization** to ensure comparability.

Section 5: Bags of Visual Words

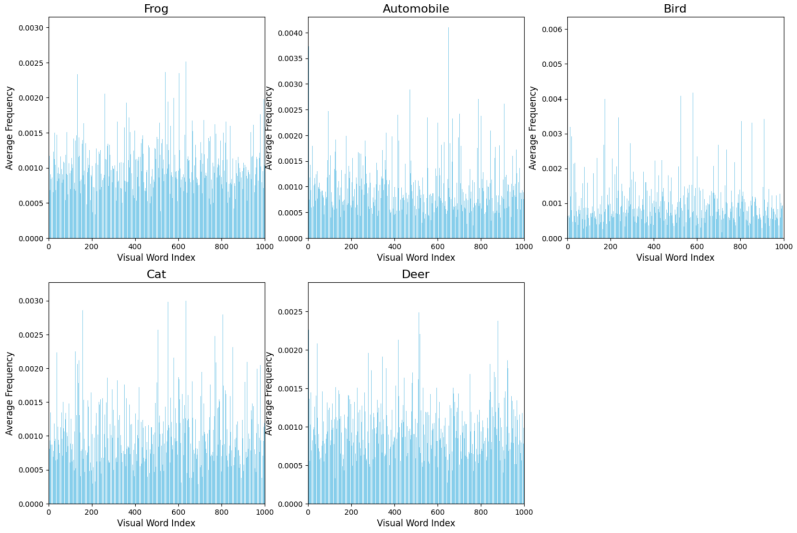


Figure 5: Bags of Visual Words for each class using SIFT extraction.

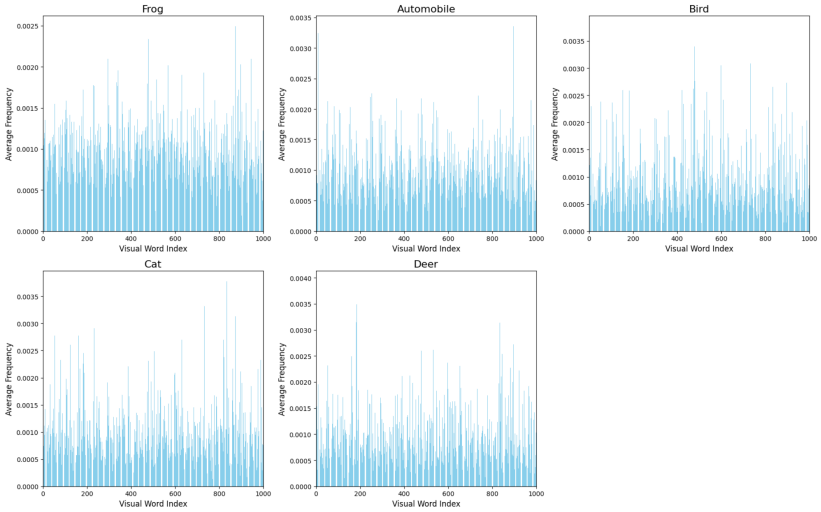


Figure 6: Bags of Visual Words for each class using ORB extraction.

Figures 5 and 6 are the bags of visual words for each class. Each bag is represented as a histogram where the frequency is the number of the descriptors, while every visual word is one of the K cluster centroids to which the descriptors were assigned. This visualization was generated after running the K-Means model with 100 iterations.

Section 6: Encoding Test Features

Similar to Section 4, the objective of this part of the project was to create representative histograms for the images within the test set. We worked with the same visual vocabularies that were created from the training data on the test images, resulting in representative histograms that could be compared with those from the training images.

Section 7 and 8: Classification and Evaluation

In this section, we used the histograms from the Bags of Visual Words to train our image classifier.

Training

For the training procedure, we took the following steps:

1. Split the data and use 50% of the images that were not used for the visual dictionary.
2. Initialize the SVM model with the following parameters:
   - `kernel = 'linear'`
   - `decision_function_shape = 'ovr'`
   - `probability = True`
   - `random_state = 42`
3. Train two versions of the model based on the feature extraction method utilized (SIFT, ORB).

Evaluation

For evaluating the models, we first classified each image and ranked them by using their confidence scores. Following this, we calculated the **Mean Average Precision (mAP)** for all classes.

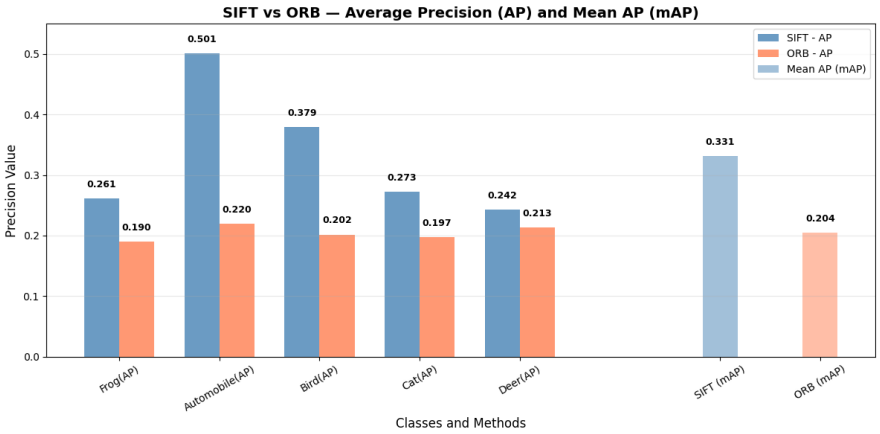
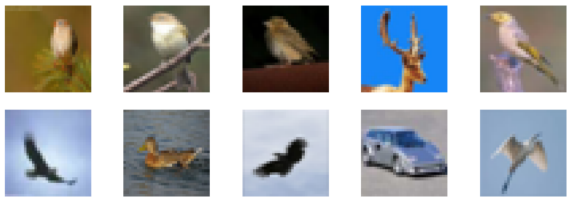


Figure 7: SIFT vs ORB (Average Precision and Mean Average Precision Comparison).

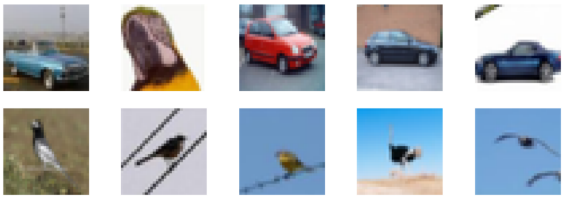
Figure 7 shows the comparison of the model performance based on the two methods of feature extraction. As we can see for both Average and Mean Average precisions, extracting features with SIFT offers better results than using ORB.

The difference in performance lies in the descriptors extracted. SIFT offers more detailed descriptors by using gradients and multiple orientations to ensure scale and rotation invariance. However, this method requires greater amounts of resources, taking longer to run as a result. ORB extracts descriptors by using binary comparison for the pixel intensities. This makes the method faster. However, it is not as robust as SIFT when it comes to scale variations. The same goes for rotation. Furthermore, SIFT uses float (128D) descriptors, which are more complex when tasked with discriminating features. Meanwhile, ORB uses binary (32D) descriptors, making discrimination less accurate.

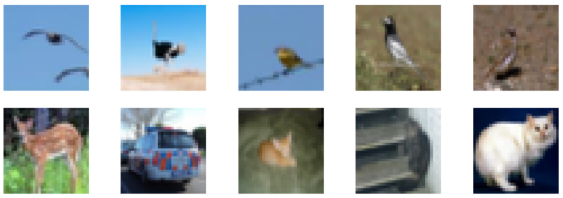
Top 5 - Bottom 5 Test Images



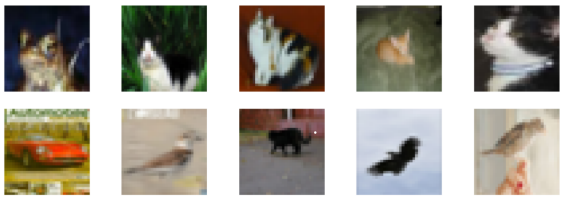
(a) Frog class



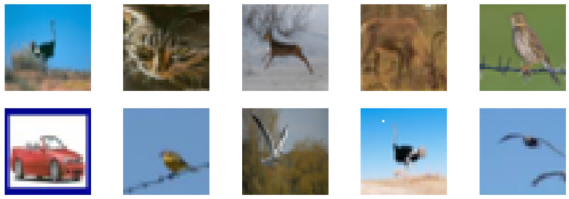
(b) Automobile class



(c) Bird class

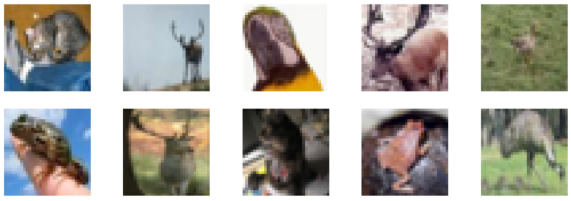


(d) Cat class

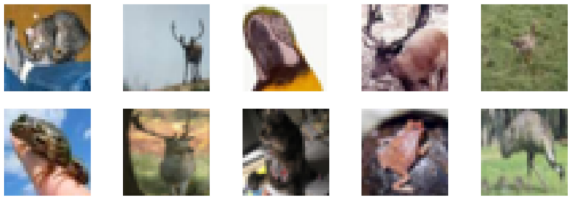


(e) Deer class

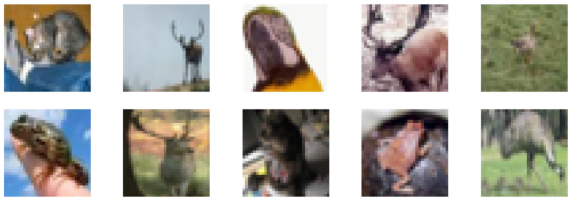
Figure 8: SIFT feature detection and matching results across CIFAR-10 classes (Top-5 and Bottom-5 test images).



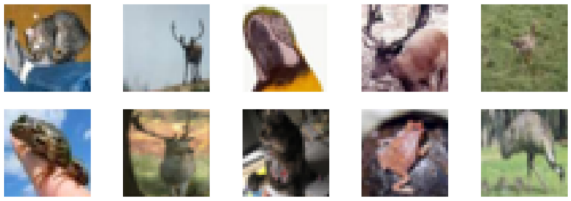
(a) Frog class



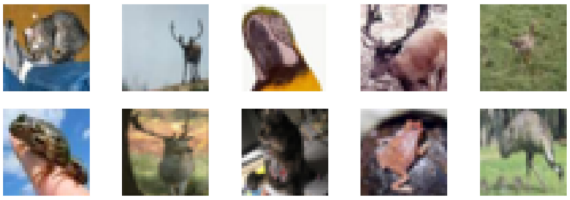
(b) Automobile class



(c) Bird class



(d) Cat class



(e) Deer class

Figure 9: ORB feature detection and matching results across CIFAR-10 classes (Top-5 and Bottom-5 test images).

Figures 8 and 9 show the Top-5 and Bottom-5 test images for every class depending on the feature extraction method. For SIFT, the classes that obtained high average precision scores, such as the automobile and bird classes, had most of their Top-5 images correctly classified. For ORB, however, we noticed that a set of images was ranked similarly despite the classes. This further proves the low precision of the classifier after using ORB as the feature extractor.

Section 9: Hyperparameter Search

The purpose of this section is to identify the optimal set of hyperparameters for the BoVW classifier, and provide a quantitative evaluation of the model's performance in the hyperparameter space. First, the hyperparameters that were used in the search are described. Second, the strategy for exploring the different configurations is explained. At last, the best performing configurations are presented.

During the hyperparameter search, an exhaustive list of parameters were evaluated and tuned:

- **Feature extraction method:** Determines the feature extraction method used for extracting keypoints from the images. Two approaches were evaluated:
  - **SIFT (Scale-Invariant Feature Transform):** Extracts scale- and rotation-invariant features based on gradient orientation histograms.
  - **ORB (Oriented FAST and Rotated BRIEF):** A computationally more efficient extractor suitable for real-time applications.
- **Vocabulary size:** Sets the number of visual words used in the construction of the visual vocabulary.
- **Subset size:** Specifies the proportion of the training dataset used for vocabulary construction. A larger value results in better representation of the training data, however increases computational complexity.
- **SVM kernel:** The kernel function used by the SVM classifier.
  - **Linear:** Computes a linear decision boundary in the feature space.
  - **RBF (Radial Basis Function):** Allows non-linear decision boundaries, useful for complex feature distributions.
- **Regularization parameter (C):** Controls the trade-off between margin maximization and classification error minimization in the SVM. Smaller values lead to stronger regularization.
- **Kernel coefficient (*gamma*):** Only suitable for RBF kernel. Defines how far the influence of a single training example reaches in the RBF kernel. The `scale` option automatically computes an appropriate value based on the input feature variance.
- **Top-N features per image:** Limits the number of features extracted from each image.

Table 2 summarizes the range of values explored during the search.

Table 2: Explored hyperparameter values

Hyperparameters	**Explored Values**
Feature extraction method	{SIFT, ORB}
Vocabulary size	{500, 1000, 1500}
Subset size	{0.3, 0.4, 0.5}
SVM kernel	{linear, rbf}
Regularization parameter (C)	{0.1, 1, 10}
Kernel coefficient (*gamma*)	{scale, 0.01, 0.001}
Top-N features per image	{50, 100, 150}

This hyperparameter grid allowed us to explore variations in feature representation, vocabulary granularity, and classifier complexity to determine the optimal configuration for the BoVW model. However, training and evaluating the total number of 648 different configurations on the entire training and test dataset requires vast computational capacities.

To mitigate the computational requirements, but also evaluate the models in a reliable manner, the following hyperparameter search strategy was followed. First, all configurations were trained on a stratified sample of 20 images per class from the training dataset (100 images in total), and evaluated on a stratified sample of 15 images per class of the test dataset. For deriving their performance, the mean Average Precision (mAP) was calculated for all configurations. Second, based on these results, the 10 best performing configurations were retained, and retrained and re-evaluated on the full training and test datasets. At last, their performance on the whole dataset was presented.

Table 3 presents the performance of the best hyperparameter configurations.

Table 3: Top 10 best-performing hyperparameter configurations based on mAP on the full dataset.

Method	**Vocabulary Size**	**Subset Frac.**	**Top-N**	**Kernel**	**C**	**Gamma**	**mAP**
SIFT	500	0.5	150	rbf	1	scale	0.4823
SIFT	1000	0.5	150	rbf	1	scale	0.4808
SIFT	1500	0.4	100	rbf	10	scale	0.4630
SIFT	500	0.3	150	rbf	1	scale	0.4659
SIFT	1000	0.3	150	rbf	1	scale	0.4676
SIFT	1000	0.4	100	rbf	10	scale	0.4421
SIFT	1000	0.5	100	rbf	0.1	scale	0.4428
SIFT	500	0.3	150	rbf	0.1	scale	0.4549
SIFT	1000	0.3	150	rbf	10	scale	0.4554
SIFT	500	0.4	150	rbf	0.1	scale	0.4529

Interpreting these results reveals several key insights. The best-performing configuration achieved a mean Average Precision (mAP) of **0.48** on the test set. This indicates that, on average, the classifier correctly retrieves the target labels roughly half of the time. Considering that this model relies solely on SIFT descriptors and histogram-based representations, this level of performance is reasonable, although it remains considerably lower than that of deep learning approaches such as convolutional neural networks (CNNs).

All of the top configurations employed the **SIFT** extractor, with the inclusion of non-linear decision boundaries through the **RBF** kernel significantly improving performance. While vocabulary size did not exhibit a clear impact, smaller vocabularies tended to produce slightly better results overall. In contrast, increasing the proportion of training data used to construct the visual vocabulary led to improved feature representations and consequently higher performance. Additionally, using a higher number of extracted features per image contributed positively to the results. Finally, examining the effect of the regularization parameter **C** showed that, although its influence was relatively minor, a medium value of C=1 generally yielded the most stable and effective performance.