Bag of Visual Words for Image Classification

Computer Vision 1 Final Lab (First Part) - University of Amsterdam, Informatics Institute

Isabelle Mudadalam (15494349), Niccolò Caselli (16391888), Marthe Schnieders (12749958)

Abstract

This study implements and evaluates a Bag of Visual Words (BoVW) image classification pipeline on five CIFAR-10 classes. Through extensive hyperparameter search, we compare SIFT and ORB feature detectors combined with K-means clustering and SVM classification. SIFT significantly outperforms ORB, achieving a maximum mAP of 0.47 compared to 0.29 for ORB. The optimal configuration uses SIFT with 500 features, 1,500 visual words, and an RBF kernel SVM.

I. Introduction

The goal of this lab assignment is to demonstrate an understanding of a basic image classification pipeline using the Bag of Visual Words (BoVW) method.

The Bag of Visual Words (BoVW) is a computer vision technique that enables image classification and retrieval by representing image features as “words.” Analogous to the Bag of Words model used in text classification, where a document is represented as a vector of word occurrence counts, the BoVW model represents an image as a histogram of visual word occurrences. Each visual word corresponds to a cluster of similar local image features extracted from training images. These clusters, which can be created with algorithms like k-means, constitute the vocabulary.

A brief overview of a typical pipeline follows:

Offline:

1) **Build a Visual Dictionary**
   Cluster local feature descriptors extracted from a training set of images to form a vocabulary of visual words.

Online (for each test image):

1) **Sample Image Patches**
   Extract patches from the test image.
2) **Extract Local Features**
   Compute feature descriptors (e.g. SIFT) for each local region sampled.
3) **Construct a Histogram**
   Assign each characteristic to the nearest visual word and build a histogram that represents the frequency of each word in the image.
4) **Classification**
   Treat the histogram as the feature vector for a machine learning classifier (e.g. SVM) to classify the test image.

II. Methodology and Implementation

This section provides a detailed overview of the design choices and experimental settings adopted in this study.

A. Dataset

The CIFAR-10 dataset [1] was used to carry out the experiments. The dataset contains 32×32 pixel RGB images, divided into ten classes, each consisting of 5,000 training images and 1,000 test images. For the purpose of this work, only the following five classes were considered:

1) Frog
2) Automobile
3) Bird
4) Cat
5) Deer

B. Feature Extraction

The feature extraction phase was conducted using two detectors for keypoint and feature extraction: SIFT (Scale-Invariant Feature Transform) [2] and ORB (Oriented FAST and Rotated BRIEF) [3]. The latter was conceived as an efficient alternative to SIFT, being rotation-invariant, resistant to noise, and two orders of magnitude faster according to the authors. The OpenCV implementations of SIFT [4] and ORB [5] were used for the experiments.

In the preliminary experimental settings, no hyperparameter tuning was performed on the detectors, except for limiting the number of features to 1000 for ORB. However, before executing the detectors, the images were upscaled by a factor of four. This step was required for the proper functioning of ORB, as discussed in the following sections.

The initial quantitative results showed that SIFT tended to detect more keypoints across all examined classes (see Table I).

From a qualitative point of view, as shown in Figures 5a and 5b in the Appendix, a clear difference between the two detector algorithms can be observed:

- While SIFT is scale-invariant and detects keypoints at multiple scales (and thus dimensions), ORB operates on fixed scales, resulting in keypoints of larger apparent size in this use case.
- SIFT produces more precise and stable keypoint detection in general.

Class	**SIFT (keypoints)**	**ORB (keypoints)**
Frog	67.82 ± 28.49	36.17 ± 24.76
Automobile	56.25 ± 22.58	34.90 ± 19.50
Bird	39.22 ± 20.57	18.32 ± 14.71
Cat	43.57 ± 22.87	22.22 ± 19.33
Deer	43.87 ± 21.62	21.37 ± 16.60

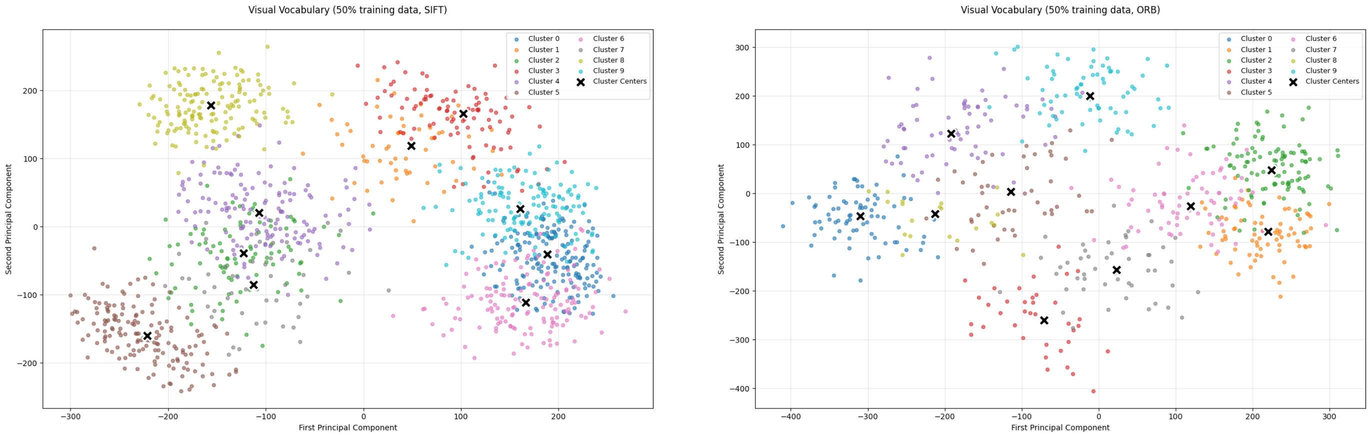
TABLE I: Comparison of SIFT and ORB keypoints statistics for different classes.

C. Visual Vocabulary Construction

For the construction of the vocabularies, the K-Means clustering algorithm was applied to the extracted features: each cluster centroid represents a visual word in the constructed vocabulary. Three different vocabularies were built for each detector, using respectively 50%, 40%, and 30% of the training dataset. For efficiency, the `MiniBatchKMeans` implementation from `scikit-learn` was employed, with a maximum of 100 iterations.

For the preliminary setting, all vocabularies were constructed using 1000 clusters (visual words); see Figure 1a and Figure 1b. The qualitative evaluation revealed that:

- **Cluster density:** SIFT clusters are denser, containing a higher number of keypoints compared to the ORB vocabulary, which exhibits sparser clusters (the total number of visual words remains the same).
- **Cluster separation:** SIFT clusters (in the 50% training setting) appear better separated in the feature space than those of ORB. This suggests that the visual words are more distinctive in the SIFT configuration, while a greater overlap between clusters can be observed in the ORB setup. Consequently, ORB seems to provide less discriminative power compared to SIFT.
- **Dataset ratio effect:** The proportion of the training dataset used for vocabulary construction appears to correlate with cluster distinctiveness. Larger subsets tend to produce more well-defined clusters, thus improving vocabulary quality.



(a) SIFT keypoints

(b) ORB keypoints

Fig. 1: Visualization of keypoints on sample images for different detectors. Vocabularies were constructed using 50% of the training data and 1000 visual words; only the first nine clusters are shown here.

D. *Feature Representation*

Each training and test image was then encoded using the six distinct vocabularies previously computed. This process involves extracting features from each image and performing a nearest neighbor search to associate each feature with the closest centroid of a visual cluster. By counting the frequency of each visual word for each image, a histogram is constructed, representing the Bag-of-Visual-Words (BoVW) encoding of the image. It is evident (see Figures 6 and 7) that SIFT histograms are less sparse than ORB histograms, which instead show few recurring words for each category class. Images 9 and 10 also depict the plotting of the mean histogram of visual words for each class, showing the distribution of visual words across the different categories in the training set.

E. *Classifier Setup*

A one-vs-rest (OvR) Support Vector Machine (SVM) classifier was employed for the classification task. An OvR classifier is a multi-class classification strategy that trains a binary classifier for each class against all other classes [6]. The classifier was trained using the remaining 50% of the training set that was not used for vocabulary construction. Various hyperparameter settings were tested, including gamma, scale, and kernel parameters (see Section III for more details).

III. Hyperparameter Search

A comprehensive grid search was conducted to identify the optimal configuration for the BoVW pipeline. The search space included parameters affecting feature extraction, vocabulary construction, and classification.

Table II summarizes all hyperparameters explored in this study. The feature extraction stage varied the detector type (SIFT vs. ORB), the number of features to extract per image, and an upscaling factor applied to input images before keypoint detection. For vocabulary construction, we explored different vocabulary sizes and training subset proportions. Finally, the SVM classifier parameters included kernel type, regularization strength (C), and the kernel coefficient (γ).

TABLE II: Hyperparameter search space

Category	**Parameter**	**Values Tested**
Feature Extraction	Detector Type	SIFT, ORB
	Number of Features	500, 1000
	Upscale Factor	2, 4
Visual Vocabulary	Number of Visual Words	500, 1000, 1500
	Training Subset Size	0.3, 0.4, 0.5
SVM Classifier	Kernel	linear, rbf
	Regularization (C)	0.1, 1.0, 10.0
	Gamma (γ)	scale, 0.01, 0.001

This parameter space resulted in a total of 1296 unique configurations. Each configuration was evaluated on the test set, and performance was measured using mean Average Precision (mAP) score.

To speed up the hyperparameter search process, the training was structured into three sequential phases:

1) **Dictionary Creation:** For each unique combination of feature extraction parameters (detector type, number of features, upscale factor) and vocabulary parameters (number of visual words, training subset size), a visual vocabulary was constructed and cached. This resulted in 72 vocabularies (2 × 2 × 2 × 3 × 3).
2) **Image Encoding:** Training and test images were encoded using each cached vocabulary.
3) **Classifier Training:** SVM classifiers with different kernel types, C values, and gamma parameters were trained on the pre-computed histogram representations.

This three-phase approach enabled substantial speedup through parallelization by leveraging Python's multiprocessing capabilities to train multiple SVM configurations simultaneously across 7 CPU cores. Thanks to this parallel execution and the caching strategy, the entire 1296 combinations search completed in approximately 100 minutes on an Apple M2 Silicon CPU.

IV. Results and Evaluation

A. Evaluation Metrics

The primary evaluation metric used in this study is **mean Average Precision (mAP)**, which provides a comprehensive measure of classification performance across all classes.

The **Precision at rank K** (P@K) is defined as:

$$P@K = \frac{TP}{TP + FP} = \frac{TP}{K} \tag{1}$$

where TP represents true positives (relevant items in the top-K results), FP represents false positives, and K is the rank threshold.

For each class c, the **Average Precision (AP@K)** is computed as:

$$AP@K = \frac{\sum_{k=1}^{K}(P@k \times \text{rel}(k))}{\text{\# relevant results}} \tag{2}$$

where $\text{rel}(k) \in \{0, 1\}$ is a relevance indicator function: $\text{rel}(k) = 1$ if the prediction at rank k belongs to class c, and 0 otherwise.

The **mean Average Precision (mAP)** is then computed by averaging the AP scores across all classes:

$$\text{mAP} = \frac{1}{C}\sum_{c=1}^{C} AP_c \tag{3}$$

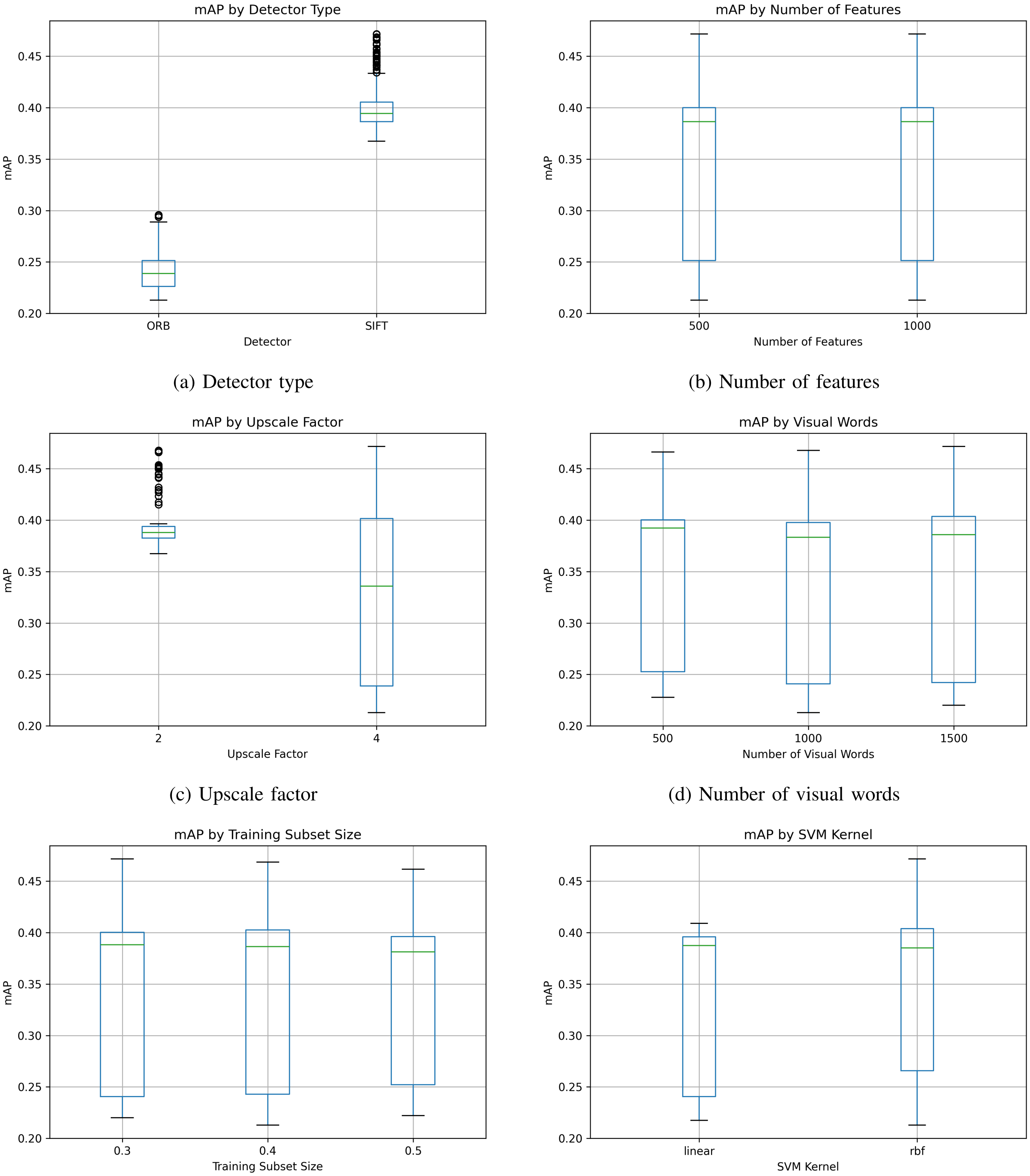
where C is the total number of classes.

B. Quantitative Results

Figure 2 shows the distribution of mAP scores for different values of each hyperparameter. The most significant findings follow:

- **Feature Detector:** SIFT consistently outperformed ORB, achieving a maximum mAP of 0.47 compared to ORB's top mAP of 0.29.
- **Number of Features:** Varying the number of features (500 vs. 1000) did not significantly affect classification performance.
- **Visual Vocabulary Size:** Changing the number of visual words (500, 1000, 1500) had little impact on mAP or its distribution. However, the 1500 visual words setup appeared in all top-5 configurations, suggesting it is optimal for peak performance.
- **Training Subset Size for Vocabulary:** Adjusting the fraction of the dataset used to create the vocabulary (0.3, 0.4, 0.5) showed no notable differences in mAP or variability.
- **SVM Kernel Choice:** The linear SVM kernel yielded more consistent mAP results, whereas the RBF kernel exhibited greater variability but was capable of achieving higher peak performance.
- **SVM Regularization (C):** The regularization parameter C had minimal effect on performance. Among the tested values, $C = 1$ demonstrated slightly better results than 0.1 and 10.
- **SVM Gamma Parameter:** The ‘scale‘ setting for the gamma parameter significantly outperformed the other tested values (0.1 and 0.01), yielding the best classification performance.

Fig. 2: Distribution of mAP scores across different hyperparameter values. Each boxplot shows the median, quartiles, and outliers for all configurations using that parameter value.



(a) Detector type

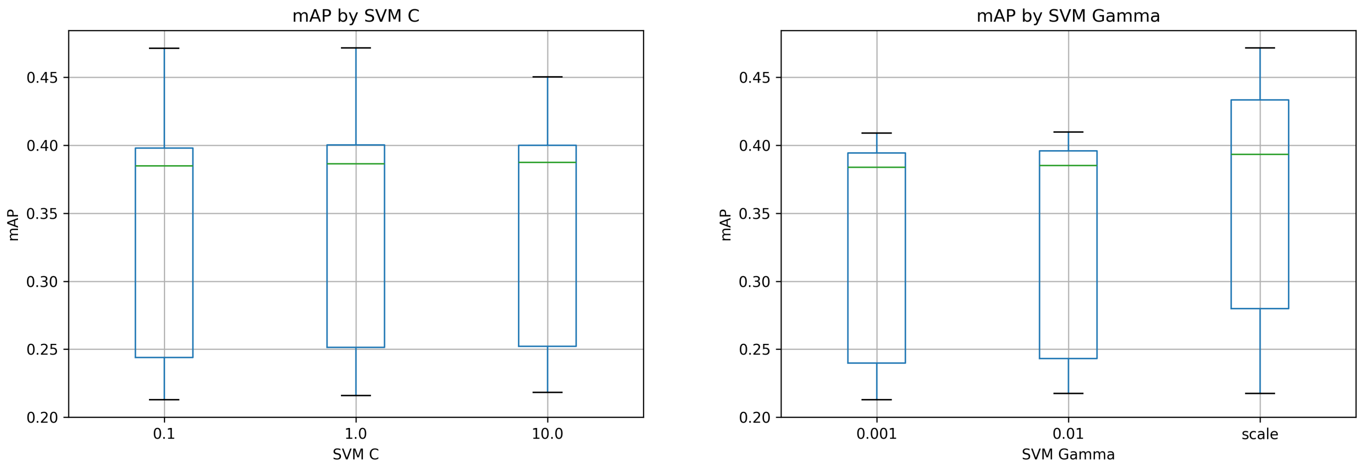
(b) Number of features

(c) Upscale factor

(d) Number of visual words

(e) Training subset size

(f) SVM kernel



(g) SVM C parameter

(h) SVM gamma parameter

Regarding the scale factor, two different values were tested: 2 and 4. SIFT, due to its inherent scale invariance, performed consistently under both settings. In contrast, ORB was unable to detect any keypoints with a scale factor of 1 or 2, and required a minimum scale factor of 4 to function correctly. Consequently, the plot of scale factor versus mAP shows a higher mAP at a scale factor of 2. This is because, at this setting, ORB fails while SIFT succeeds, and SIFT alone provides a higher overall mAP than ORB.

To sum up, Figure 3 and Table III show the top-5 configurations ranked by mAP. All top-performing configurations use SIFT as the feature detector. The best configuration is characterized by 500 features, 1500 visual words, 30% of the training set for vocabulary creation, an RBF kernel, $C = 1$, and SVM$_\gamma$ = scale. Figure 4 and Table IV show the worst-performing configurations.

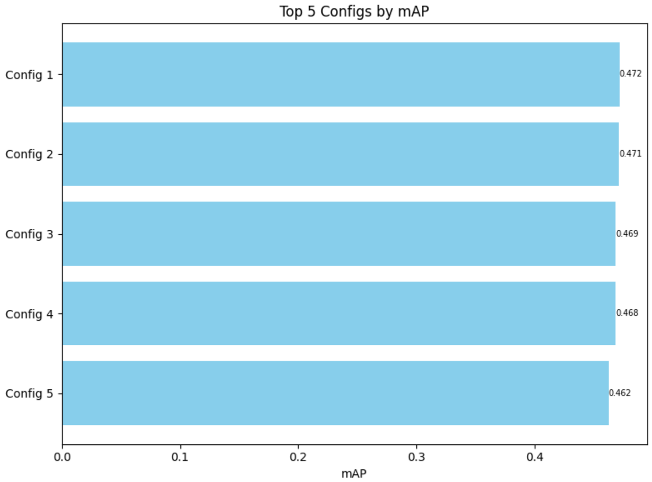


Fig. 3: Performance visualization of top 5 configurations.

TABLE III: Top 5 configurations by mean Average Precision (mAP)

Det.	nF	Up	Words	Sub.	Ker.	C	γ	mAP	Acc.
SIFT	500	4	1500	0.3	rbf	1.0	scale	0.4716	0.482
SIFT	500	4	1500	0.3	rbf	0.1	scale	0.4714	0.479
SIFT	500	4	1500	0.4	rbf	1.0	scale	0.4685	0.454
SIFT	500	4	1500	0.4	rbf	0.1	scale	0.4685	0.459
SIFT	500	4	500	0.3	rbf	0.1	scale	0.4624	0.458

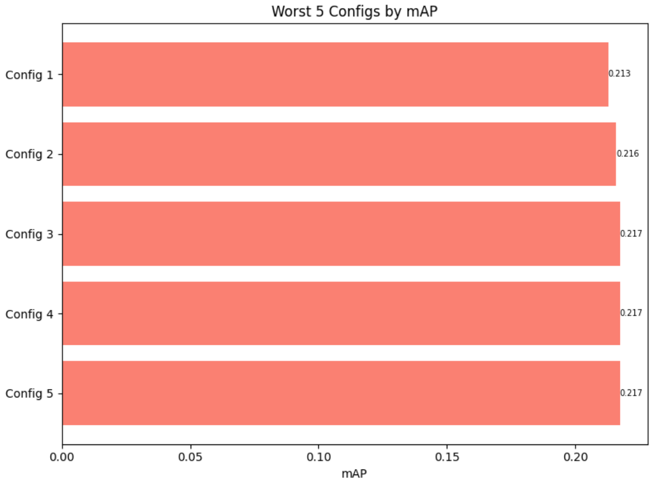


Fig. 4: Performance visualization of worst 5 configurations.

TABLE IV: Worst 5 configurations by mean Average Precision (mAP)

Det.	nF	Up	Words	Sub.	Ker.	C	γ	mAP	Acc.
ORB	500	4	1000	0.4	rbf	0.1	0.001	0.2129	0.222
ORB	500	4	1000	0.4	rbf	1.0	0.001	0.2159	0.228
ORB	500	4	1000	0.4	linear	0.1	scale	0.2173	0.220
ORB	500	4	1000	0.4	linear	0.1	0.01	0.2173	0.220
ORB	500	4	1000	0.4	linear	0.1	0.001	0.2173	0.220

C. Qualitative Results

To gain deeper insight into the behavior of different feature detectors, we can qualitatively examine some samples of the top-5 and bottom-5 ranked images by mAP for each class. In this report, for brevity reasons, only three classes were displayed: automobile, cat, and frog, as we argue they are the most representative. The figures are displayed in the appendix. The setup used in this qualitative evaluation consists of a linear SVM with a vocabulary built from 50% of the training data and 1000 visual words.

Starting with SIFT: the automobile class (Figure 8a) is representative because it is a class whose top-5 images are all correct (full precision for the top-5, and overall AP: 62.09%). Probably, the effectiveness of the classification is due to sharp geometric edges (wheels, windows, body panels). For the cat class (Figure 8b), the overall AP was 32.02%; we can, in fact, appreciate that the images in the second to fifth positions in the ranking are not cats. Finally, the frog class achieves the lowest precision with SIFT (26.73%), and visually, we can see that the top-3 predictions are all wrong (deer and cats are mistaken for frogs). This underlines how SIFT struggles with organic textures. In fact, as already shown in Table I, the frog is the class that yields the highest number of features (probably due to the texture, prominent eyes, etc.).

With respect to ORB, the retrieved images indicate consistently low precision across all classes, suggesting a fundamental limitation in the representational capacity of the vocabulary.

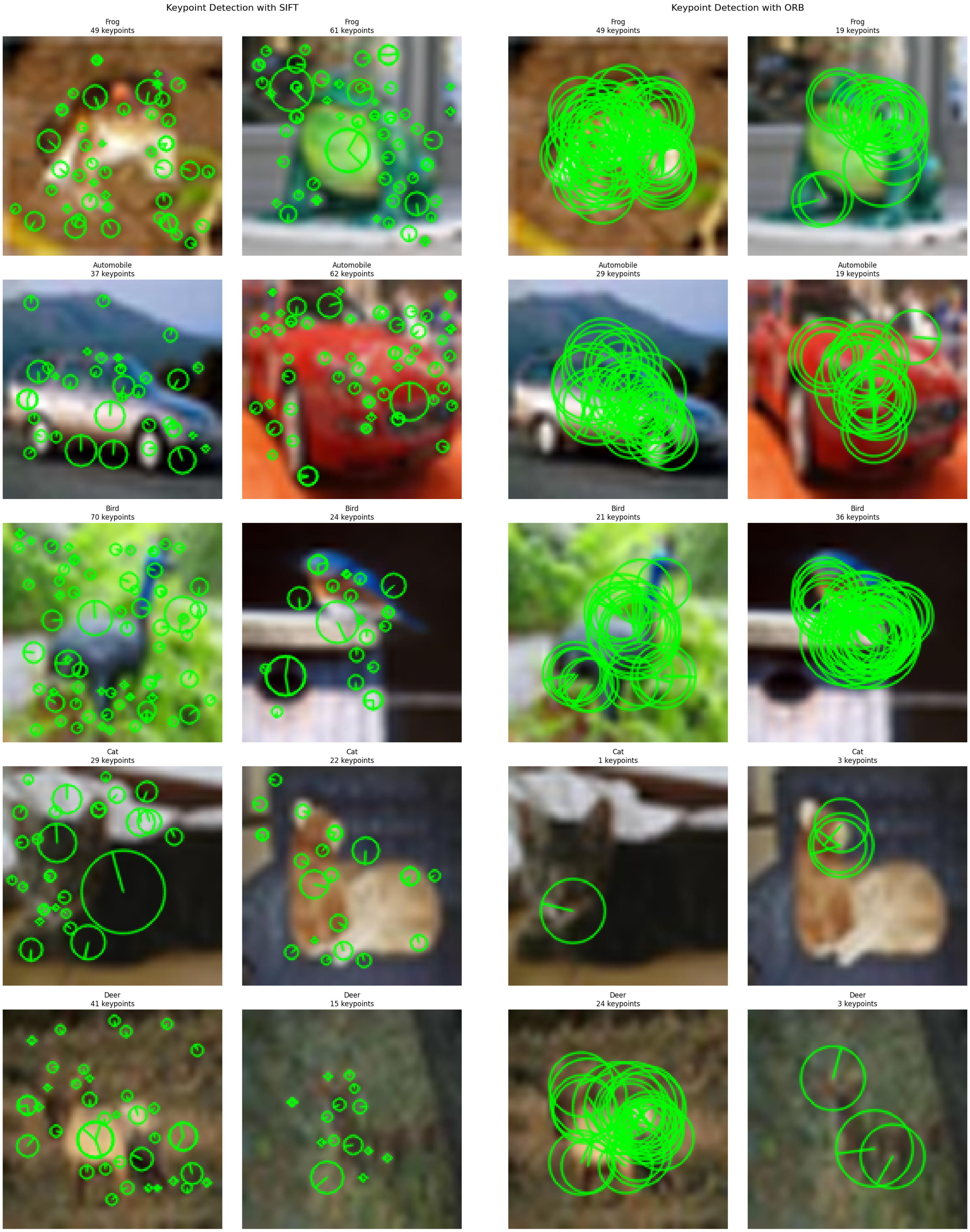
V. Conclusion

In this study, we implemented an image classification setup based on the Bag of Visual Words (BoVW) technique. Through hyperparameter search, we observed that the choice of feature detector has the most significant impact on classification performance: SIFT consistently outperformed ORB, owing to its inherent scale invariance and more distinctive keypoint representation; ORB, in contrast, struggled to detect keypoints at lower scale factors, requiring image up-scaling.

The optimal configuration utilized SIFT with 500 features per image, a vocabulary of 1500 visual words constructed from 30% of the training data, and an RBF kernel SVM with $C = 1$ and $\gamma =$ scale. Interestingly, vocabulary size showed a modest impact on overall precision, though larger vocabularies appeared in all top-5 configurations. Similarly, the fraction of training data used for vocabulary construction had minimal effect on classification accuracy.

These results validate the BoVW framework's ability to capture visual patterns through local feature aggregation. However, the modest performance (mAP $\approx$ 0.47) highlights the limitations of hand-crafted features, suggesting that more sophisticated representations or deep learning approaches (e.g, visual transformers like CLIP) may be necessary.

APPENDIX



(a) SIFT keypoints

(b) ORB keypoints

Fig. 5: Comparison of SIFT and ORB keypoint detection

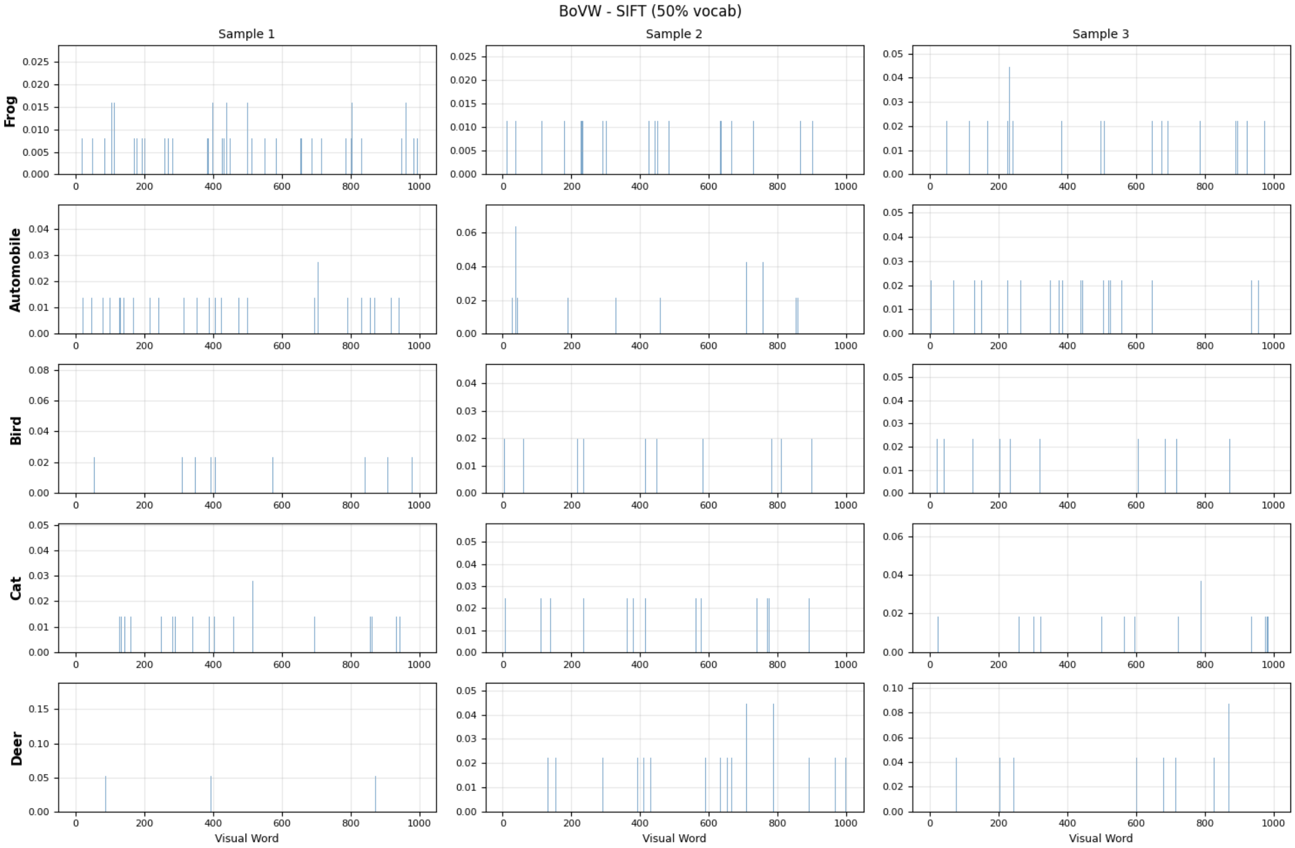


Fig. 6: SIFT histogram with 50 visual words

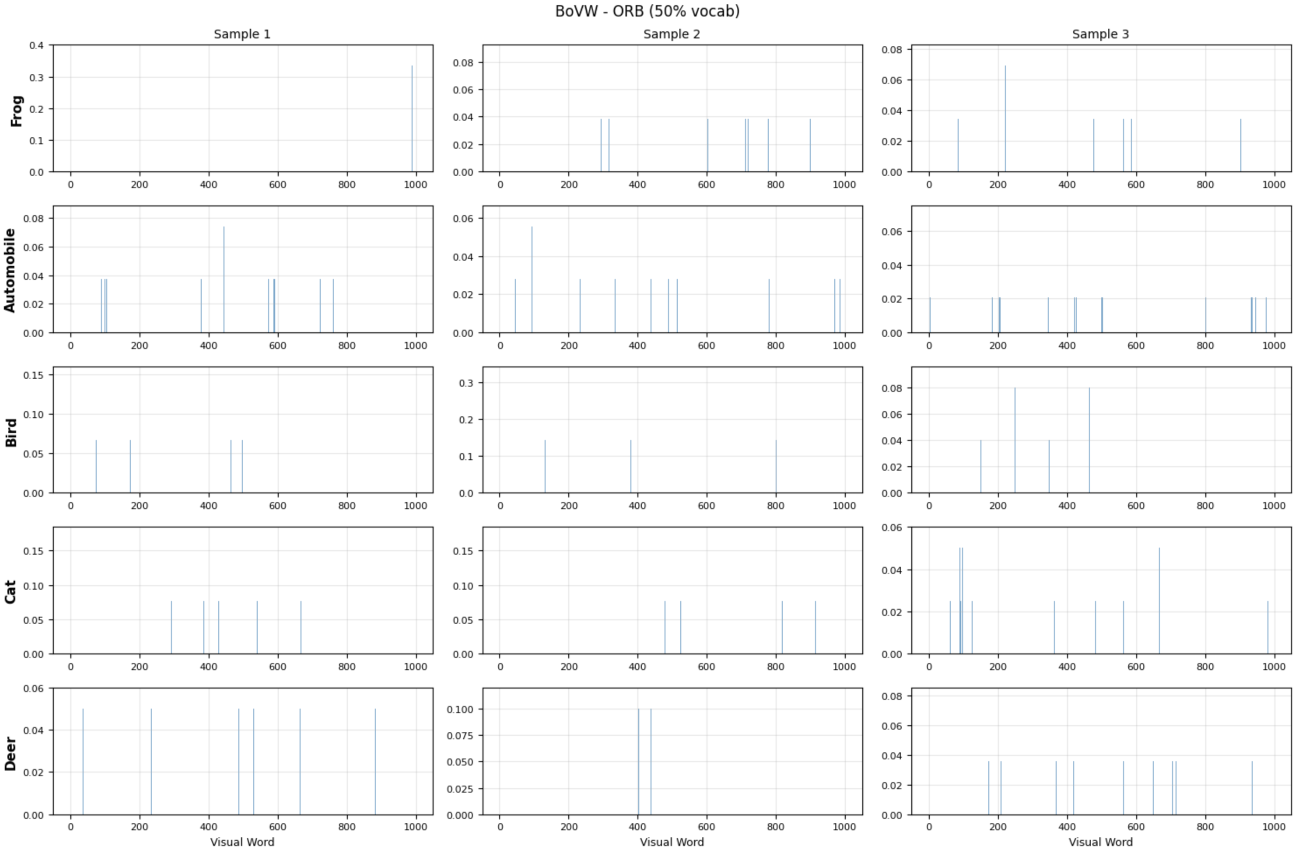
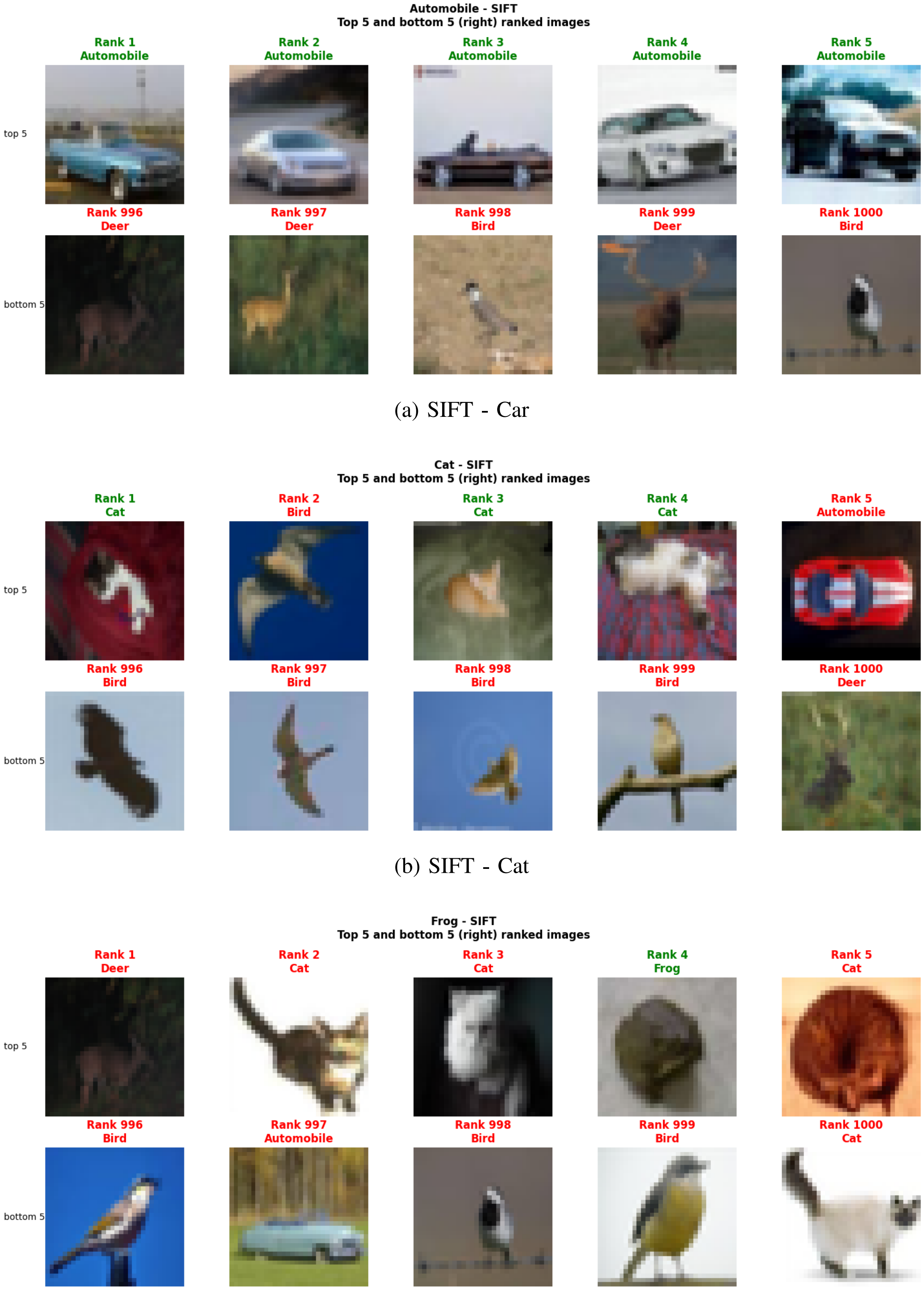


Fig. 7: ORB histogram with 50 visual words

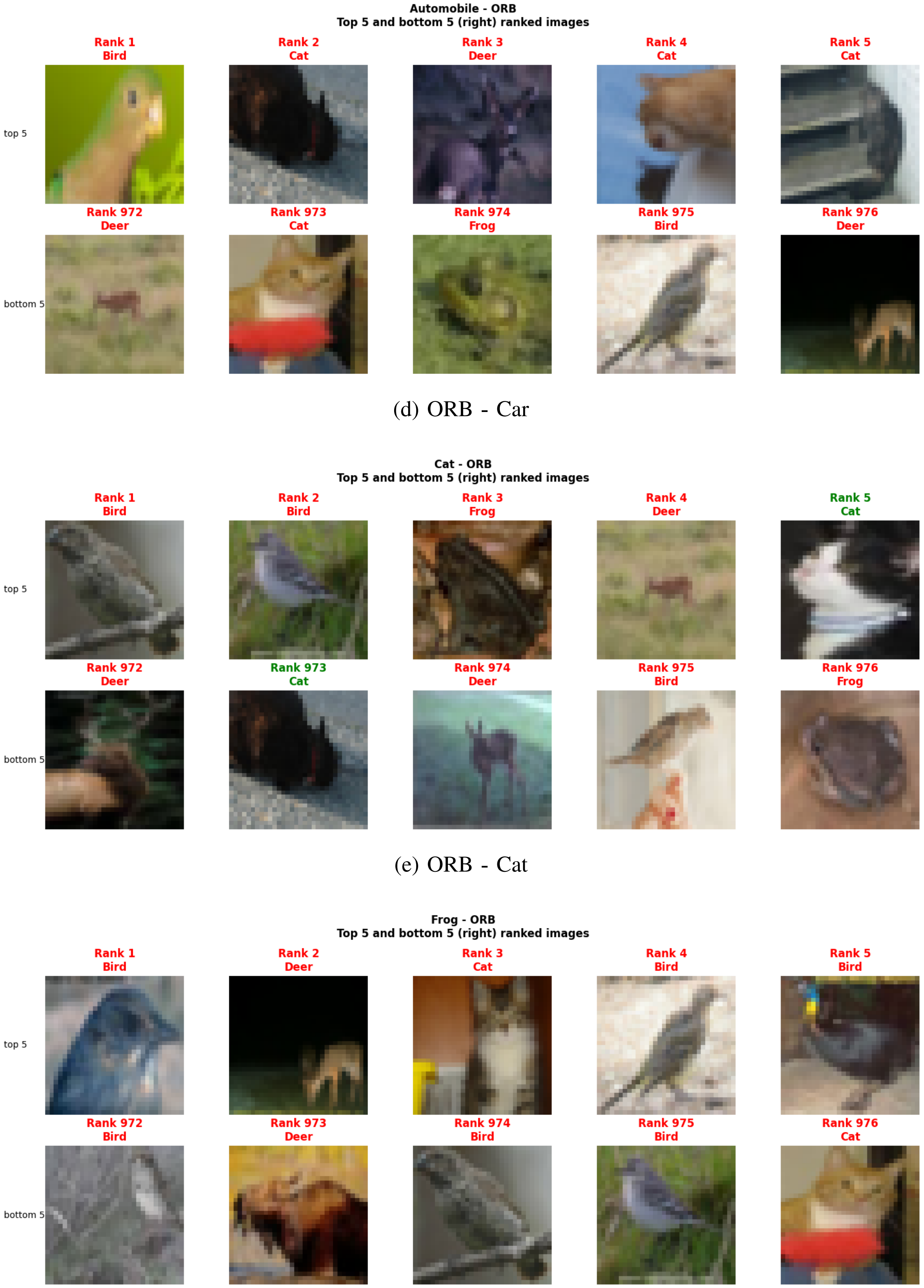


(a) SIFT - Car

(b) SIFT - Cat

(c) SIFT - Frog

Fig. 8: Top-5 and bottom-5 ranked images by class (with SIFT)



(d) ORB - Car

(e) ORB - Cat

(f) ORB - Frog

Fig. 8: Top-5 and bottom-5 ranked images by class (with ORB)

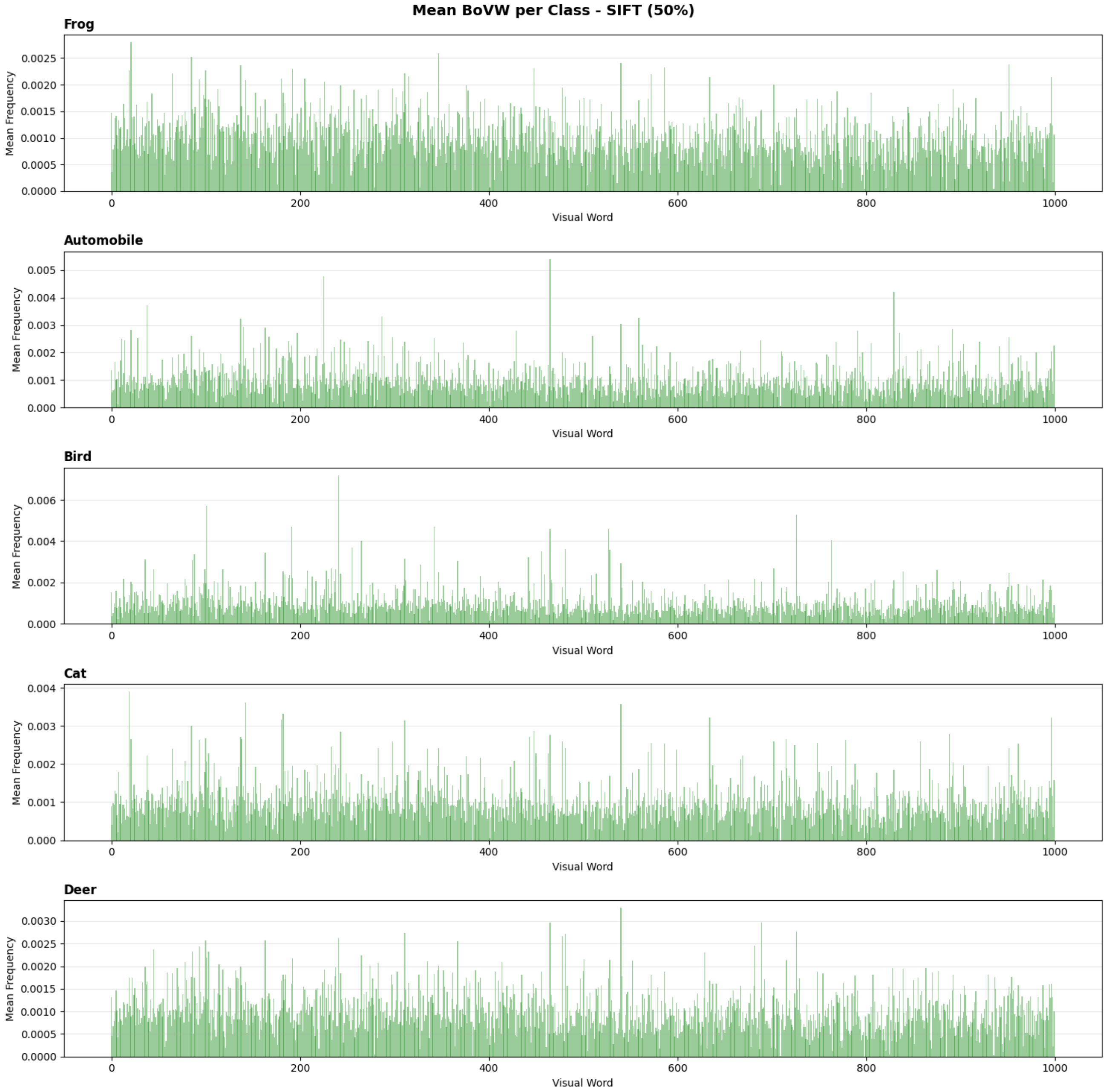


Fig. 9: Mean BoVW with SIFT

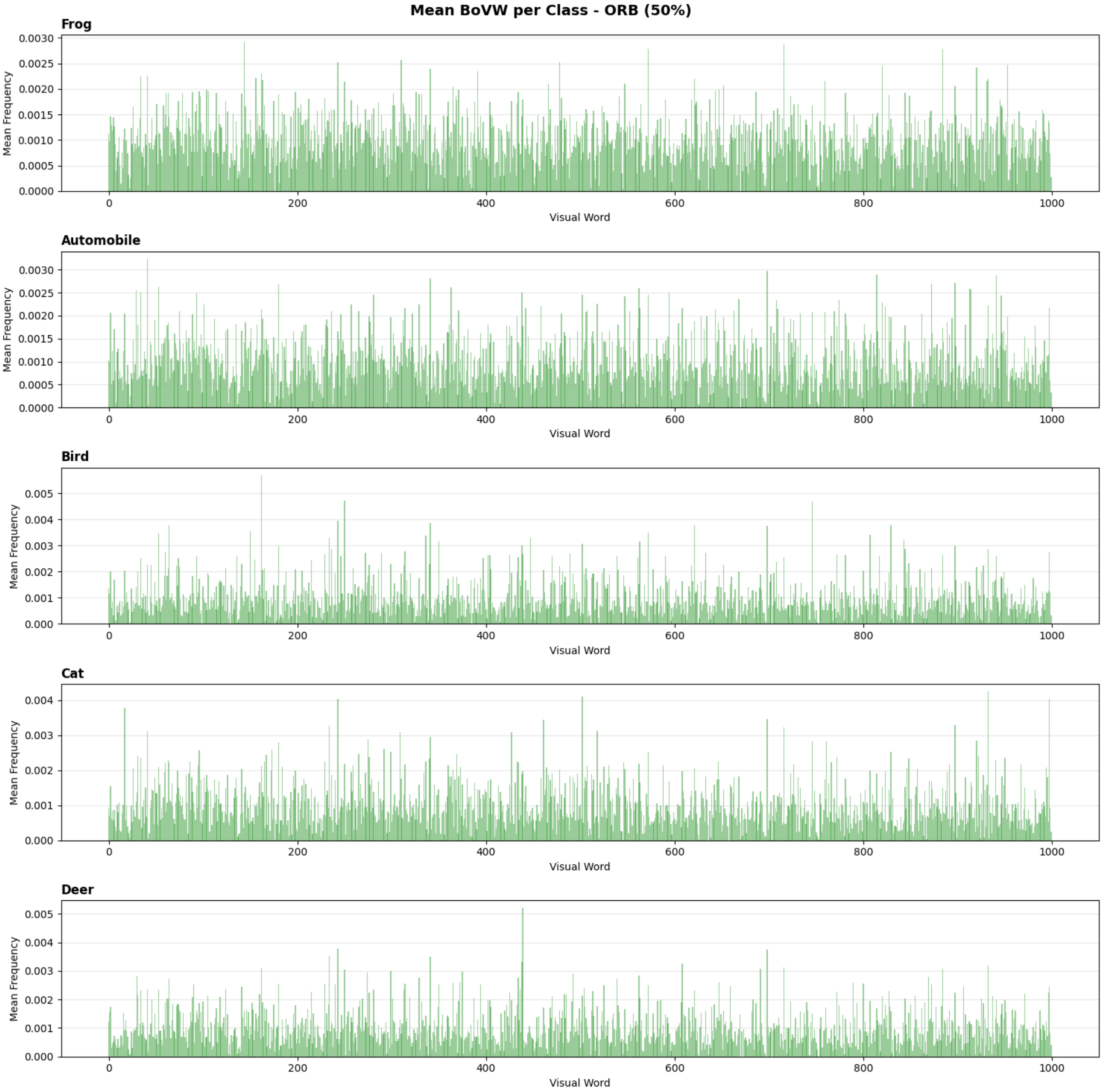


Fig. 10: Mean BoVW with ORB

REFERENCES

[1] A. Krizhevsky, “Learning multiple layers of features from tiny images,” tech. rep., University of Toronto, Toronto, Canada, 2009.

[2] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2564–2571, IEEE, 2011.

[4] OpenCV Documentation Team, “Introduction to sift (scale-invariant feature transform).” https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html, 2025. Accessed: 2025-10-19.

[5] OpenCV Documentation Team, “Orb (oriented fast and rotated brief).” https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html, 2025. Accessed: 2025-10-19.

[6] scikit-learn developers, “sklearn.multiclass.onevsrestclassifier.” https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html, 2025. Accessed: 2025-10-19.