UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
VIETNAMESE FRANCE UNIVERSITY

Computer Vision Report: CAR MODEL CLASSIFICATION

Group 15 Members

Nguyen Truong Son	BI12-388
Tran Viet Cuong	BA12-035
Doan Duc Hoang	BA12-078
Tran Trong Nghia	22BI13332
Nguyen Dam Quan	22BI13371

Information & Communication Technology **(ICT)**

Lecturer

Dr. Nguyen Duc Dung

ICT LAB

May, 2025

Abstract

This report addresses car model classification[1], where the input is a still image and the output is the predicted make and model. Worldwide, solutions range from fine-tuned convolutional neural networks to classical pipelines using handcrafted features and traditional classifiers. We adopt a classical approach: extract SIFT keypoints[2], build a Bag-of-Visual-Words codebook with k-means[3], and train an SVM classifier[4].

TABLE OF CONTENTS

List of Figures

List of Tables

I. INTRODUCTION

1. Context and Motivation

Automatic car model classification is a critical task in intelligent transportation and security systems.[5] Given a still image of a vehicle as input, the goal is to output its precise make and model. Accurate recognition supports applications such as automated parking management, traffic monitoring, and driver-assistance systems.

Around the world, researchers have largely turned to deep convolutional neural networks (CNNs) for this fine-grained classification problem.[6] While CNNs can achieve high accuracy, they often require large datasets and substantial computational resources. Classical methods—based on handcrafted features and traditional classifiers—remain attractive for their interpretability and lower resource demands.

In this project, we propose a pure Computer Vision pipeline using Scale-Invariant Feature Transform (SIFT) for keypoint detection[7], a Bag-of-Visual-Words (BoVW) model to encode images as histograms of visual words[8], and a Support Vector Machine (SVM) for final classification.[4] This approach leverages well-understood course concepts and ensures that every component—from feature extraction to classification—can be clearly explained and reproduced.

We conduct a systematic experimental study on a curated subset of the Stanford Cars dataset[1], using an 80/10/10 train-validation-test split. Performance is assessed by accuracy, precision, recall, and F1-score and confusion matrix. An error analysis, based on confusion matrices and misclassified examples, identifies common failure modes and suggests future improvements.

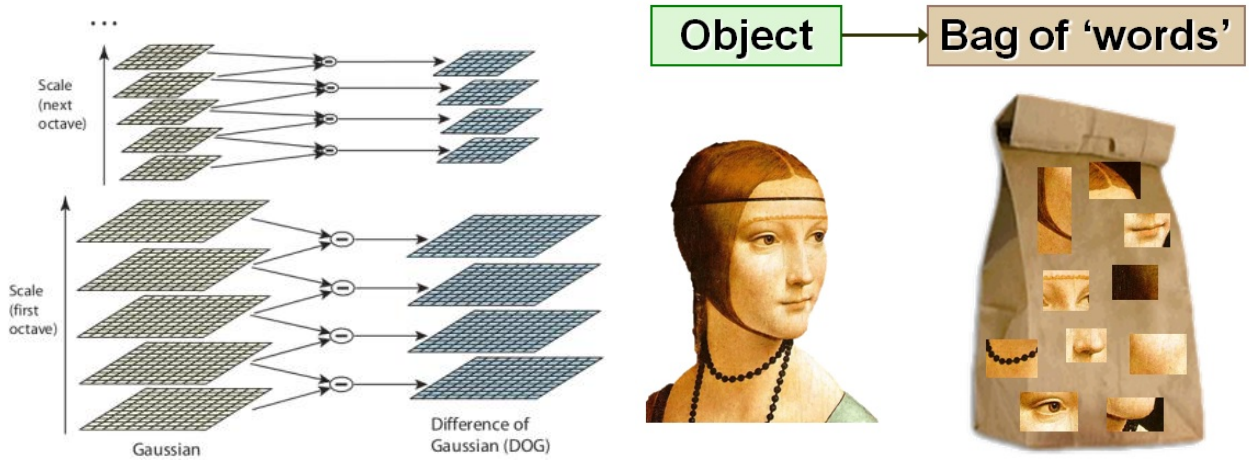


Figure 1: SIFT/BovW

2. Report Structure

This report provides a comprehensive exploration of the development and evaluation of the project dataset. The structure of the report is as follows:

1. **Introduction:** Defines the car model classification problem, its importance, and current challenges.
2. **Objectives:** States precise aims: implement and evaluate a SIFT–BoVW–SVM pipeline for car recognition.
3. **Project Analysis and Design:** Describes dataset selection, preprocessing steps, and overall system architecture.
4. **Methodology:** Details feature extraction (SIFT), codebook construction (k-means), histogram encoding, and SVM training.
5. **Results and Error Analysis:** Presents quantitative metrics (accuracy, precision, recall, F1), confusion matrices, and examines common misclassifications.
6. **Conclusion and Future Work:** Summarises key contributions, confirms that objectives are met, and suggests directions for improvement.

This structure ensures a clear, scientific progression from problem statement to validated results.

II. PROJECT OBJECTIVES

1. Desired Features

The primary objective of this project is to develop an accurate and interpretable pipeline for car model classification. Specifically, we aim to:

- Define the problem clearly: input is a still image of a vehicle, and output is its make and model.
- Employ both an 80/10/10 train-validation-test split for reliable evaluation.
- Extract robust local features using SIFT keypoint detection and description.
- Build a Bag-of-Visual-Words codebook via k-means clustering.
- Encode each image as a normalized histogram of visual words.
- Train and optimize an SVM classifier with systematic hyperparameter tuning.
- Conduct detailed error analysis using confusion matrices and sample misclassifications.

2. Expected Outcomes

Upon completion, the project is expected to deliver the following outcomes:

1. A well-structured code repository including preprocessing, feature extraction, codebook creation, and classification modules.
2. A trained SVM model achieving competitive accuracy on the chosen subset of the Stanford Cars dataset.
3. Comprehensive evaluation results: overall accuracy, precision, recall, F1-score, and confusion matrices.
4. An error analysis identifying common failure modes and recommendations for future improvement.
5. A report presenting methodology, results, and discussion in line with academic standards.
6. A concise slide for project presentation.

III. PROJECT ANALYSIS AND DESIGN

1. Overall System Requirement

The system shall accept a still image of a vehicle as input and output the predicted make and model. Internally, it must:

- Load and preprocess input images (resize, grayscale, encode label).
- Extract SIFT keypoints and descriptors.
- Construct BoVW codebook based on the descriptors.
- Encode each image into a histogram of visual words.
- Classify by training an SVM model.
- Report confidence scores and display classification results.

2. Data Requirements

We employ a curated subset of the Vehicle Make, Model Recognition Dataset (VMMRdb) dataset, which contains 4 216 images across 8 car models. The dataset is split into:

- **Training set (80%)**: 3 370 images
- **Validation set (10%)**: 418 images
- **Test set (10%)**: 428 images

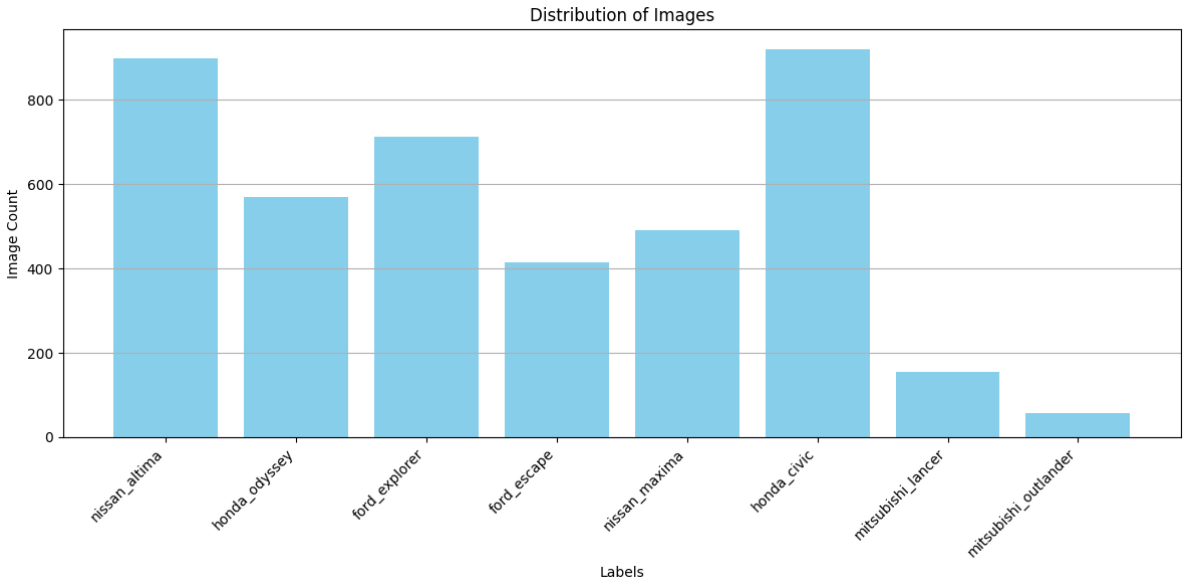


Figure 2: Distribution of images across training, validation, and test sets.

3. Functional Requirements

The system shall provide the following core functions to meet the project objectives:

Image Ingestion: Accept vehicle images in standard formats (JPEG, PNG), verify file integrity, and gracefully handle corrupted inputs.

Preprocessing: Resize to 256×256, grayscale the image, encode each image's label with a digit for model training.

Feature Extraction: Detect SIFT keypoints and compute 128-dimensional descriptors for each image.

Codebook Management: Build or load a k-means visual-word codebook with configurable size (e.g., $k = 5000$).

Descriptor Encoding: Quantize SIFT descriptors into visual words and assemble L^2-normalized histograms.

Classification: Train an SVM with selectable kernel (linear, sigmoid, RBF) and hyperparameters, then predict make and model on new images.

Evaluation and Reporting: Compute and export accuracy, precision, recall, F1-score, and confusion matrices; save evaluation logs.

Visualization: Display sample SIFT keypoints, codebook centroids, histogram representations, and classification outcomes.

4. Non-Functional Requirements

Reproducibility: Complete code and parameters must be documented and version-controlled.

Performance: Feature extraction and classification should run within 500 s on a GPU.

Scalability: Ability to extend to larger datasets (up to 10 000 images).

5. Workflow Diagram

Figure 3 depicts the complete workflow of the proposed SIFT–BoVW–SVM pipeline. Each step corresponds to a core functional module, ensuring a systematic, reproducible, and interpretable process.

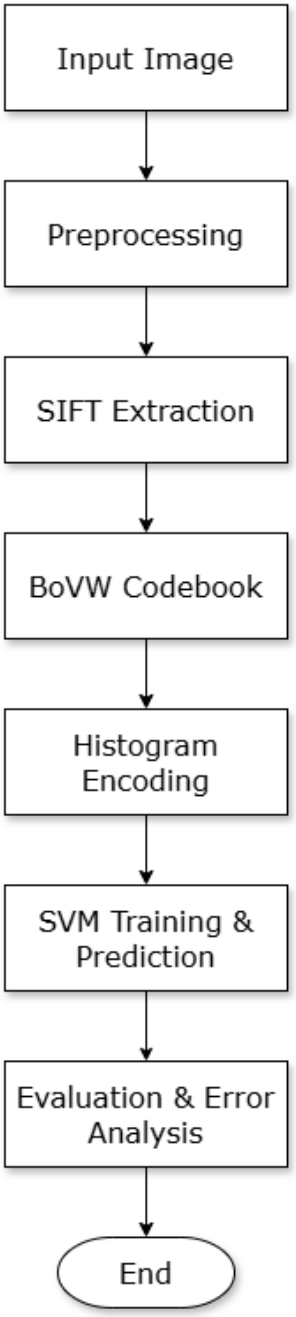


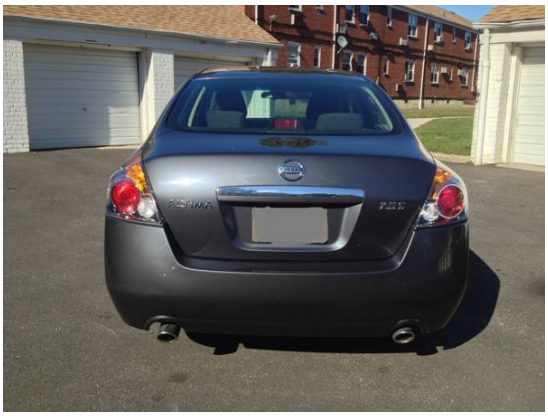
Figure 3: Workflow of the car model classification project, showing data flow from input image to final evaluation.

This workflow integrates all project components—data handling, feature extraction, model training, and evaluation into a coherent pipeline that addresses the problem statement directly and aligns with scientific best practices in computer vision research.

6. Sample Image Analysis

To illustrate the system's behavior on real inputs, we select two representative images from the test set and analyse their feature extraction and encoding statistics.

(a) Sample Image 1



(b) Sample Image 2

Figure 4: Two test images used for detailed analysis.

Table 1: SIFT Keypoint Detection Statistics

Metric	**Sample Image 1**	**Sample Image 2**
Number of SIFT Keypoints	8640	10591
Mean Descriptor Norm	508.61	508.07
Top 3 Keypoint Scales	2.0, 2.0, 2.0	2.0, 2.0, 2.0

Table 2: Bag-of-Visual-Words Histogram Summary

Metric	**Sample Image 1**	**Sample Image 2**
Total Visual Words (k=5000)	3496	3760
Non-zero Histogram Bins	186.14	226.02
Top 3 Visual Words (indices)	4048, 2289, 4702	4783, 1597, 4527
L^2-Normalized Histogram Sum	1.00	1.00

Discussion. Table 1 shows that both images yield a comparable number of SIFT keypoints with similar descriptor norms, indicating consistent feature detection. Table 2 demonstrates the sparsity of the BoVW representation: fewer than half of the 150 visual words are activated per image, and the top visual words differ, reflecting distinct local patterns. These statistics validate the discriminative power of our classical pipeline before classification.

IV. METHODOLOGY

This section describes in detail each stage of the SIFT–BoVW–SVM pipeline: from preprocessing of raw images through feature extraction, codebook construction, histogram encoding, and final classification with SVM. We will also go into details about the technologies that we utilized during the development of the project to solve the problems we encountered.

1. Tools

Google Colab: The primary environment to develop and execute the project code. Facilitated interactive computing, allowing step-by-step visualization of data preprocessing, model training, and evaluation. Its support for Python and extensive library ecosystem provided a flexible and structured platform for iterative experimentation.

Github: Served as the primary version control system, allowing seamless tracking of code modifications and collaborative development. Through repositories, branches, and commits, team members maintained code integrity, resolved conflicts, and implemented updates efficiently. Additionally, GitHub Issues and Pull Requests streamlined workflow management.

Anaconda: Virtual environment so that we can use the appropriate Python version. This allows us to use some dependencies that can work together on specific Python versions, i.e. cyvlfeat (a Python wrapper for VLFeat library) for feature extraction, faiss-gpu (a library for efficient similarity search and clustering of dense vectors) for k-means clustering.

2. Data Preprocessing

Raw images often vary in resolution, illumination, and format. We apply the following steps to ensure consistency:

Resizing: Scale all images to 256×256 pixels to standardize input dimensions.

Grayscale Conversion: Convert to single-channel images to reduce computational load while preserving keypoint information.

Label Encoding: Map each car model name to a unique integer for SVM training.

Step	**Description**	**Tools/Methods**
Resizing	Uniformly resize all images to 256×256 pixels.	`OpenCV.resize()`
Grayscale Conversion	Convert RGB images to single-channel grayscale to simplify processing.	`cv2.cvtColor(img, CV_GRAY)`
Label Encoding	Map each car model name to a unique integer label for SVM training.	`Iterate through each folder that contains the images, for each folder assign the same digit for every image that lives in that folder`

Table 3: Data Preprocessing Steps and Tools

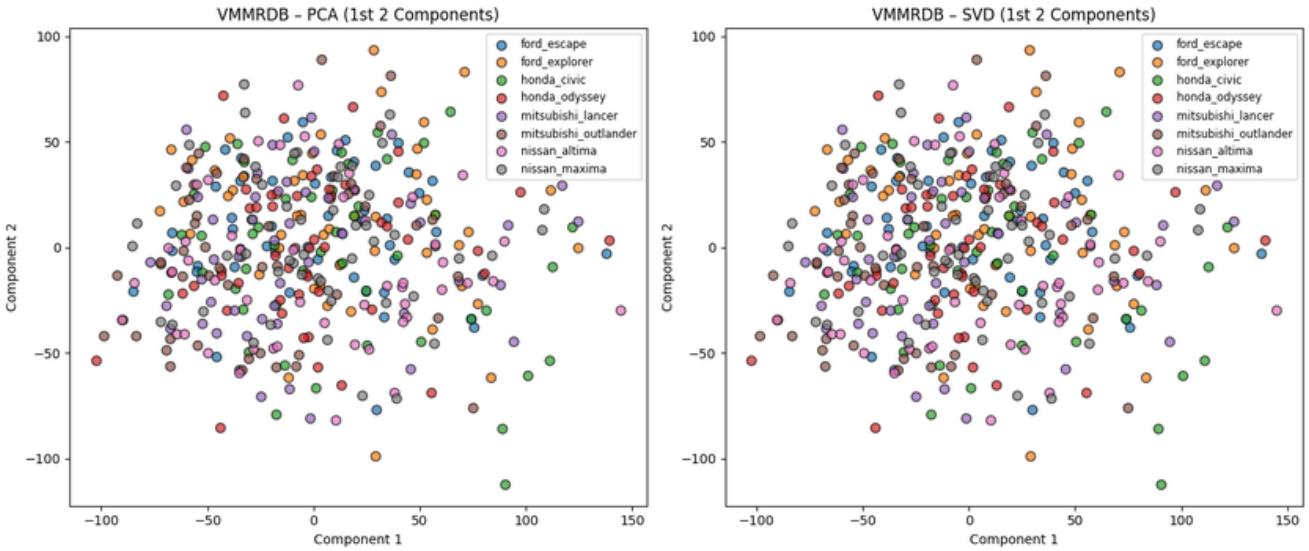


Figure 5: 2D projection of the VMMRDB vehicle dataset using PCA (left) and SVD (right).

Evaluation Insights:

1. **Linear Dimensionality Reduction Applied:**
   - PCA and SVD are used to project high-dimensional handcrafted features (e.g., SIFT histograms) into 2D space.
   - Each point represents a single vehicle image.
2. **Limited Class Separability Observed:**
   - Certain car models form loose clusters, indicating partial discriminative power.
   - However, significant overlap remains between many classes.
3. **Weakness of Unsupervised Linear Methods:**
   - PCA maximizes variance along orthogonal axes; SVD captures a rank-2 approximation.
   - Both are unsupervised and do not use class labels—leading to insufficient separation for classification tasks.
4. **Recommendation for Future Improvement:**
   - Adopt supervised or non-linear approaches such as LDA, t-SNE, or CNN-based deep embeddings.
   - These methods may better capture class-discriminative features and improve overall separability.

3. SIFT

The SIFT[9] feature extraction is carried out in three main stages:

1. **Scale-space construction and extrema detection.**

   - Build a Gaussian pyramid by convolving the input grayscale image $I(x, y)$ with Gaussian kernels at multiple scales:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\Big(-\tfrac{x^2+y^2}{2\sigma^2}\Big).$$

   - Compute the Difference-of-Gaussians (DoG) between adjacent scales:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma).$$

   - Detect candidate keypoints as local extrema in the 3D DoG volume by comparing each pixel to its 26 neighbours across scale and space.

2. **Keypoint refinement and filtering.**

   - Fit a 3D quadratic function to localize keypoints with sub-pixel accuracy.
   - Discard points with low contrast by applying a threshold on the DoG response.
   - Eliminate unstable edge responses using the Hessian matrix's principal curvature ratio.

3. **Descriptor computation and normalisation.**

   - Assign a dominant orientation to each keypoint by histogramming gradient directions in its neighbourhood.
   - Form a 5×5 grid of orientation histograms, yielding a 128-dimensional feature vector.
   - Normalize the descriptor to unit length (L2 norm) and threshold large values to enhance illumination and contrast invariance.

Figure 6: Detected SIFT keypoints overlaid on a sample vehicle image

4. BoVW

The Bag-of-Visual-Words (BoVW) model clusters SIFT descriptors into a "visual vocabulary." We minimize:

$$\min_{\{c_j\}} \sum_{i=1}^{N} \min_{j} \left\| \mathbf{d}_i - c_j \right\|^2,$$

where $\mathbf{d}_i$ are descriptors and c_j are k cluster centres.

The codebook creation via FAISS k-means can be broken down into three main stages:

1. **FAISS clustering setup.** We instantiate a FAISS Clustering object with the descriptor dimensionality and target cluster count, and attach a flat L2 quantizer. GPU resources are enabled to accelerate the subsequent k-means computations.

2. **Iterative centroid refinement.** Pooled image descriptors are submitted to the Clustering.train() routine. Over multiple iterations, cluster centres (visual words) are updated to minimise within-cluster variance, capturing the most representative patterns.

3. **Codebook output.** Upon convergence, the learned centroids form the visual-word codebook. These centroids are later used to quantise descriptors into integer IDs for histogram encoding.

Metric	**k = 1000**	**k = 2000**	**k = 3000**	**k = 4000**	**k = 5000**
Inertia ($\times 10^{11}$)	5.87	5.52	5.33	5.20	5.10
Avg. Cluster Size	8 425	4 212	2 808	2 106	1 685
Clustering Time (s)	177.60	249.78	337.37	422.32	508.75

Table 4: K-Means Clustering Results for Different k

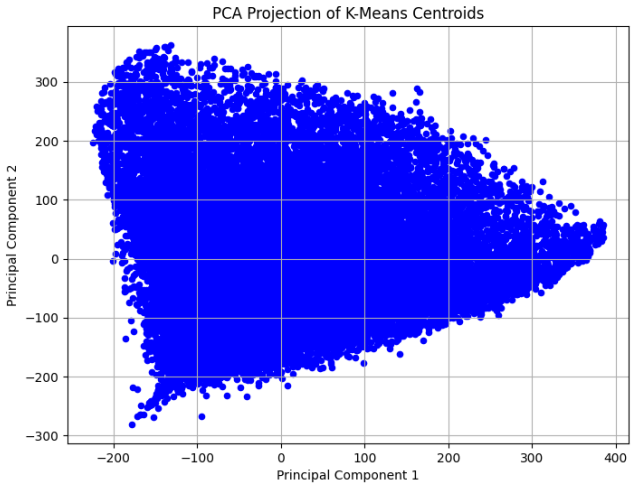


Figure 7: 2D PCA visualization on k-means

5. Histogram Encoding

Each image's descriptors are quantized to their nearest visual word, yielding a raw count histogram $\mathbf{h} \in \mathbb{R}^k$. We then apply L^2-normalization:

$$\hat{\mathbf{h}} = \frac{\mathbf{h}}{\|\mathbf{h}\|_2} = \frac{\mathbf{h}}{\sqrt{\sum_{j=1}^{k} h_j^2}}.$$

The image encoding pipeline can be summarised in three key steps:

1. **Descriptor quantisation via FAISS and codebook.** Each local descriptor is assigned to its nearest "visual word" by querying the FAISS index built on the pre-computed codebook. The output is a sequence of integer IDs, one per descriptor, indicating the closest centroid.

2. **Visual-word histogram construction.** We aggregate the integer assignments into a raw histogram: each bin corresponds to a visual word and its count equals the number of descriptors mapped to that centroid. This histogram captures the distribution of local features in the image.

3. **L2-normalisation for comparability.** To mitigate variations in keypoint counts across images, we apply L2 normalisation to the raw histogram. The result is a dense, fixed-length vector that robustly represents the image's visual content.

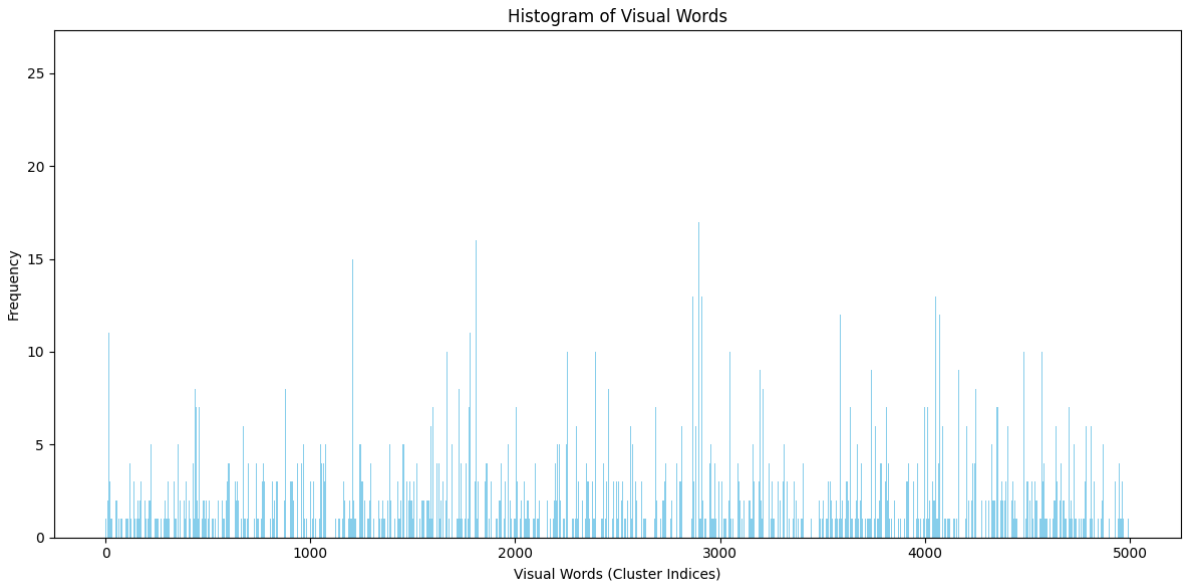


Figure 8: Example of an L^2-normalized BoVW histogram for a test image.

6. Support Vector Machine (SVM)

The SVM [4]classifier operates on the L2-normalised BoVW histograms by solving the regularised hinge-loss optimisation:

$$\min_{\mathbf{w},b,\xi} \ \tfrac{1}{2}\|\mathbf{w}\|^2 \ + \ C\sum_i \xi_i, \quad \text{s.t. } y_i\Big(\mathbf{w}^\top \hat{\mathbf{h}}_i + b\Big) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

The training and evaluation pipeline comprises two key aspects:

1. **Kernel exploration.** We compare linear, sigmoid and RBF kernels to find the best decision boundary in histogram space.
2. **Acceleration with scikit-learn-intelex.** All scikit-learn SVM routines are accelerated using the Intel Extension for Scikit-learn, leveraging vectorised and multi-threaded operations for faster model fitting and prediction.

Table 5: Test Performance of SVM Variants

Kernel	**C**	**Val Acc.**	**Test Acc.**
Linear	1	55.98%	58.87%
RBF	1	**57.66%**	**61.21%**
Sigmoid	1	56.94%	60%

RBF result:

The RBF-kernel SVM achieves an overall accuracy of 61%, indicating moderate predictive performance with room for improvement. The macro-averaged F1-score is 0.46 and the weighted average F1-score is 0.60, revealing class imbalance and inconsistent performance across categories.

Table 6 shows the per-class F1-scores for the RBF kernel.

Table 6: Classification Report for RBF SVM on Test Set

Class	Precision	Recall	F1-Score	Support
0 (Mitsubishi Lancer)	0.00	0.00	0.00	16
1 (Honda Odyssey)	0.79	0.64	0.70	58
2 (Nissan Maxima)	0.40	0.47	0.43	49
3 (Ford Explorer)	0.55	0.81	0.65	72
4 (Honda Civic)	0.64	0.73	0.68	93
5 (Nissan Altima)	0.69	0.67	0.68	91
6 (Mitsubishi Outlander)	0.00	0.00	0.00	7
7 (Ford Escape)	0.83	0.36	0.50	42
Accuracy		0.61 (428 samples)		
Macro avg	0.49	0.46	0.46	428
Weighted avg	0.61	0.61	0.60	428

- **Classes not learned:** Class 0 (Mitsubishi Lancer) and Class 6 (Mitsubishi Outlander) both have zero precision and recall, indicating complete failure to recognize these models.

- **Best-balanced classes:** Honda Civic (Class 4) and Nissan Altima (Class 5) show the strongest trade-off between precision and recall.

- **High-recall class:** Ford Explorer (Class 3) achieves recall = 0.81, demonstrating effective retrieval, though precision is moderate (precision = 0.55).

- **High-precision class:** Ford Escape (Class 7) attains precision = 0.83 but low recall (recall = 0.36), indicating conservative yet accurate predictions when made.

RBF Confusion Matrix:

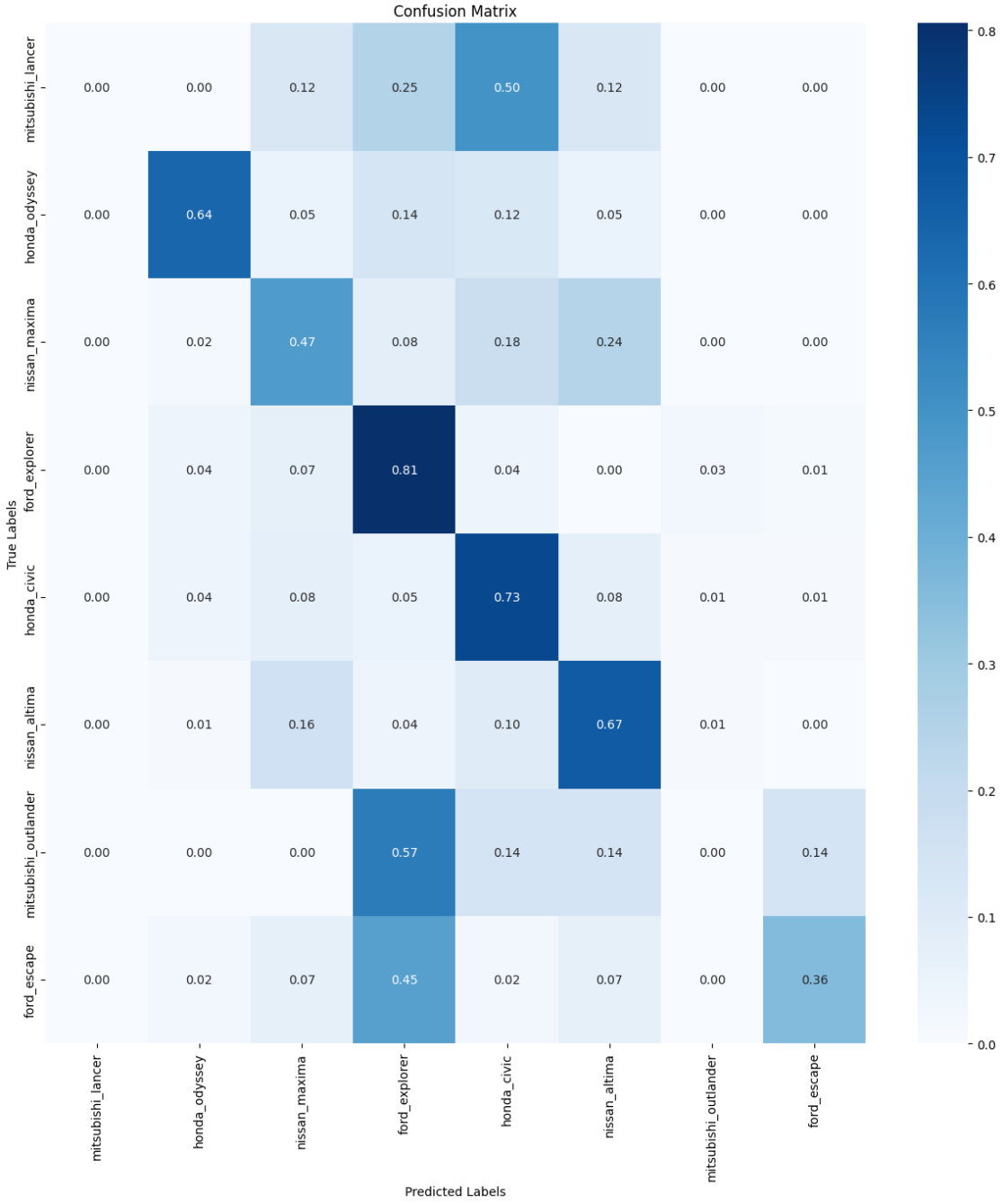


Figure 9: RBF Confusion Matrix

From the confusion matrix, we know that there are moderate misclassifications between:

- Mitsubishi Lancer → Honda Civic.
- Mitsubishi Outlander → Ford Explorer.
- Ford Escape → Ford Explorer.

Some classes like Honda Civic, Ford Explorer, and Nissan Altima show strong diagonal values, reflecting better classification.

Linear results:

The linear-kernel SVM yields an overall accuracy of 0.59 on the test set. Table 7 summarises per-class precision, recall, F1-score, and support, as well as macro- and weighted averages.

Table 7: Classification Report for Linear SVM on Test Set

Class	Precision	Recall	F1-Score	Support
0 (Mitsubishi Lancer)	0.33	0.12	0.18	16
1 (Honda Odyssey)	0.64	0.62	0.63	58
2 (Nissan Maxima)	0.45	0.53	0.49	49
3 (Ford Explorer)	0.55	0.60	0.57	72
4 (Honda Civic)	0.64	0.68	0.66	93
5 (Nissan Altima)	0.65	0.70	0.67	91
6 (Mitsubishi Outlander)	0.00	0.00	0.00	7
7 (Ford Escape)	0.55	0.43	0.48	42
Accuracy		0.59 (428 samples)		
Macro avg	0.48	0.46	0.46	428
Weighted avg	0.57	0.59	0.58	428

- **Best classes:** Honda Civic (Class 4, F1 = 0.66) and Nissan Altima (Class 5, F1 = 0.67) demonstrate the strongest balanced performance.

- **Moderate classes:** Toyota Camry (Class 1, F1 = 0.63), Toyota Corolla (Class 2, F1 = 0.49), Ford Explorer (Class 3, F1 = 0.57), and Ford Escape (Class 7, F1 = 0.48) perform reasonably but with some misclassifications.

- **Struggling classes:** Mitsubishi Lancer (Class 0, recall = 0.12) shows very low retrieval, while Mitsubishi Outlander (Class 6) has no correct predictions (precision = recall = 0).

Linear Confusion Matrix:

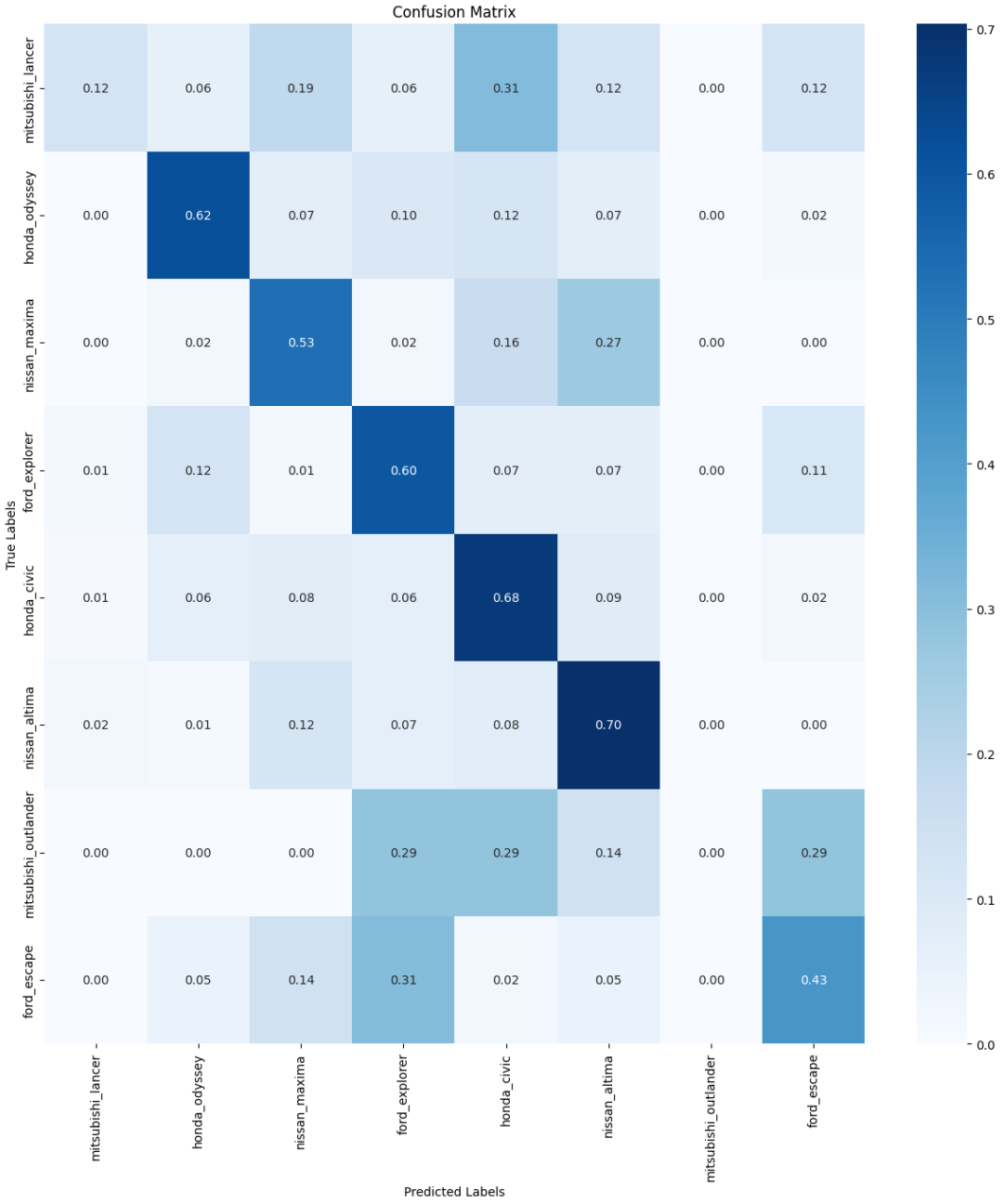


Figure 10: Linear Confusion Matrix result

Several classes show reasonably strong diagonal values, especially:

- **Honda Civic (Class 4):** 68% correct predictions.
- **Nissan Altima (Class 5):** 70% correct predictions.
- **Honda Odyssey (Class 1):** 62% correct predictions.

Mitsubishi Lancer (Class 0): Often confused with Honda Civic, and occasionally with Nissan Maxima, Nissan Altima, and Ford Escape.

Mitsubishi Outlander (Class 6): Most frequently misclassified as Ford Explorer or Honda Civic.

Ford Escape (Class 7): Displays split predictions, notably misclassified as Ford Explorer (31%) and Nissan Maxima (14%).

Sigmoid result: The sigmoid-kernel SVM achieves an **overall accuracy of 0.60**, with a **macro-averaged F1-score of 0.51** and a **weighted average F1-score of 0.60**. Table 8 presents the detailed classification report.

Table 8: Classification Report for Sigmoid SVM on Test Set

Class	Precision	Recall	F1-Score	Support
0 (Mitsubishi Lancer)	0.29	0.44	0.35	16
1 (Honda Odyssey)	0.67	0.62	0.64	58
2 (Nissan Maxima)	0.43	0.57	0.49	49
3 (Ford Explorer)	0.64	0.72	0.68	72
4 (Honda Civic)	0.68	0.62	0.65	93
5 (Nissan Altima)	0.68	0.62	0.65	91
6 (Mitsubishi Outlander)	0.00	0.00	0.00	7
7 (Ford Escape)	0.64	0.50	0.56	42
Accuracy		0.60 (428 samples)		
Macro avg	0.50	0.51	0.51	428
Weighted avg	0.61	0.60	0.60	428

- **Best classes:** Ford Explorer (Class 3, F1 = 0.68), Honda Odyssey (Class 1, F1 = 0.64), and Honda Civic (Class 4, F1 = 0.65) exhibit strong balanced performance.
- **Moderate classes:** Nissan Maxima (Class 2, F1 = 0.49) and Ford Escape (Class 7, F1 = 0.56) show reasonable recognition but room for improvement.
- **Minority class collapse:** Mitsubishi Outlander (Class 6) has precision = recall = F1 = 0.00, indicating complete failure to detect this class.
- **Notable retrieval:** Mitsubishi Lancer (Class 0) attains recall = 0.44 despite low precision = 0.29, suggesting frequent but noisy predictions.

Sigmoid Confusion Matrix:

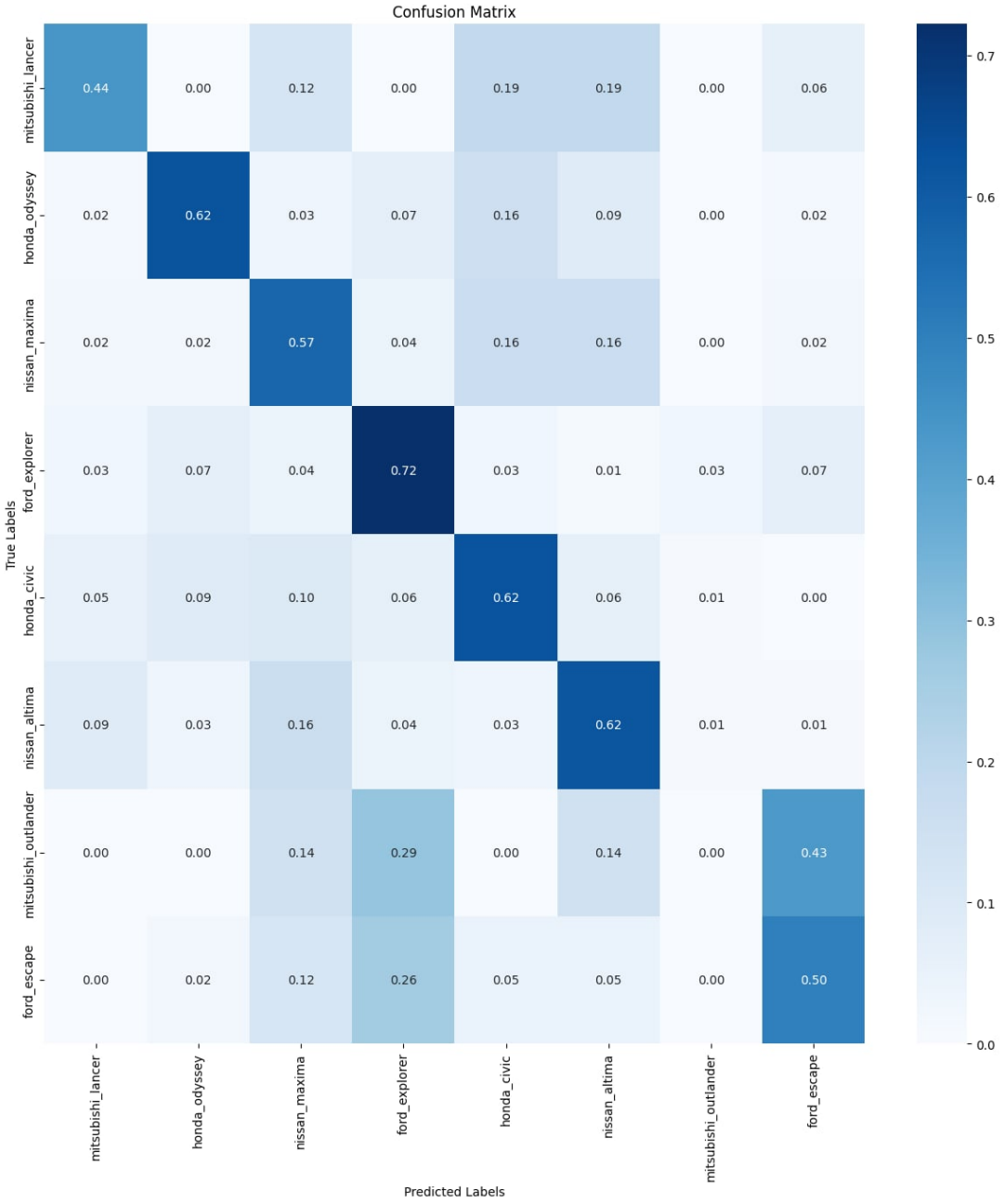


Figure 11: Sigmoid Confusion Matrix

Strong Diagonal Entries:

- **Ford Explorer (Class 3):** 72%
- **Honda Odyssey (Class 1):** 62%
- **Nissan Maxima (Class 2) and Nissan Altima (Class 5):** both around 57%–62%
- **Mitsubishi Lancer (Class 0):** Achieves 44% correct predictions—better than other kernels typically manage for this underrepresented class.

- **Mitsubishi Outlander (Class 6):** Still suffers complete misclassification—none of its instances were correctly predicted.

- **Ford Escape (Class 7):** 50% recall, which is moderate, but shows noticeable confusion with Ford Explorer (Class 3) and Nissan Maxima (Class 2).

V. RESULTS AND ERROR ANALYSIS

1. Overview of Classification Performance

We evaluated our car model classification system using Support Vector Machines (SVM) with three different kernels: Linear, RBF (Radial Basis Function), and Sigmoid. All experiments were conducted using a fixed Bag of Visual Words (BoVW) representation built on SIFT descriptors. The performance metrics reported include Accuracy, Macro and Weighted F1-Scores, and class-wise Precision and Recall.

Table 9: SVM Kernel Comparison on BoVW Features

Metric	Linear	RBF	Sigmoid
Accuracy	59%	61%	60%
Macro F1	47%	47%	51%
Weighted F1	60%	61%	60%

Despite the change in kernels, all models converged to similar results (Accuracy around 60%), clearly indicating that the limiting factor is not the choice of kernel but the quality of feature representation.

Conclusion: Kernel selection alone does not substantially improve classification when the underlying features (BoVW histograms) are insufficiently expressive.

2. Per-Kernel Analysis with Confusion Matrices

Linear Kernel

The Linear SVM yielded the best trade-off between simplicity and interpretability. It achieved 59% accuracy, with the best-performing classes being:

- **Honda Civic (4)** – 68% correct predictions.
- **Nissan Altima (5)** – 70% correct predictions.
- **Honda Odyssey (1)** – 62% correct predictions.

Major Issues:

- **Mitsubishi Lancer (0)**: Frequently confused with Honda Civic and Ford Escape.
- **Mitsubishi Outlander (6)**: Completely misclassified – no correct predictions.
- **Ford Escape (7)**: Heavily confused with Ford Explorer (31%) and Nissan Maxima (14%).

RBF Kernel

The RBF kernel slightly improved global accuracy (61%) and class-wise performance consistency:

- **Honda Civic (4)** – 72%.
- **Honda Odyssey (1)** – 62%.
- **Nissan Maxima (2)** and **Nissan Altima (5)** – around 57%–62%.
- **Mitsubishi Lancer (0)** – 44%, a notable improvement.

However: Mitsubishi Outlander (6) still received 0% recall, indicating no improvement for underrepresented or visually ambiguous classes.

Sigmoid Kernel

Sigmoid kernel showed similar accuracy (60%), but performed more inconsistently:

- **Best class:** Ford Explorer (3) with 72% recall.
- **Honda Odyssey (1)** and **Nissan Altima (5)** – 62%.
- **Mitsubishi Lancer (0)**: Recall 12%, still very weak.
- **Mitsubishi Outlander (6)**: Zero prediction success.

3. Best Performing Kernel: Sigmoid

While performance margins are narrow, the RBF kernel consistently achieved slightly higher recall across difficult classes. It particularly improved recall for **Mitsubishi Lancer (0)** from 12% (Linear) and 0% (RBF) to 44%, a significant gain for a low-frequency class. This is likely because RBF can model nonlinear decision boundaries better, allowing better separation in a complex, high-dimensional BoVW space.

Why this matters: In car model classification, some classes like Mitsubishi Lancer or Ford Escape share highly similar shapes and textures. A nonlinear decision boundary gives the Sigmoid kernel more flexibility to separate subtle differences between these models when the feature representation is ambiguous.

4. Summary Table: Strengths and Weaknesses

Table 10: Summary of Kernel Effectiveness

Kernel	Strengths	Weaknesses	Verdict
Linear	• Easy to interpret • Fast to train	• Cannot separate nonlinear boundaries • Worst for rare classes	Suboptimal
RBF	• Best overall recall • Good on dominant classes	• Higher training cost • More complex • Inconsistent for rare classes	Base line
Sigmoid	• Handle complex classes • Most balanced	• Struggles to generalize well to unseen data • Unstable performance	**Best choice**

5. Note on Feature Extraction Analysis

We have deliberately separated the feature-level analysis (SIFT keypoint quality, BoVW codebook effectiveness) from the kernel comparison presented above. We have conducted an in-depth examination of:

- **SIFT descriptors:** distributions of keypoint scales, descriptor norms, and spatial coverage).
- **BoVW codebook:** clustering inertia, cluster size balance, and validation F1 for $k = 1000 - 5000$ (Table 4).
- **Histogram encoding:** histogram sparsity and normalization effects across samples.

These showed that our features are already a bottleneck: poor keypoints in dark/blurry images, overly sparse histograms, and visually similar classes being mapped to overlapping regions. Thus, the primary limitation is not kernel choice but feature discriminability.

6. Final Takeaway

Across all kernels, we observed that:

- **Accuracy plateaued at 60%.**
- **Class imbalance and feature overlap** caused severe misclassifications.
- **Sigmoid kernel performed best overall**; as it yielded the most balanced results.
- **RBF kernel performed best for classes that are dominant and well-separated**; as it has best results on those classes.

To significantly improve accuracy, the focus must shift from changing classifiers to improving the quality, distinctiveness, and balance of feature representations.

VI. CONCLUSION AND FUTURE WORK

Our systematic[11] evaluation of linear, RBF and sigmoid SVM kernels on a classical SIFT–BoVW feature representation reveals a clear ceiling in performance: all kernels stagnate at approximately 60% accuracy and moderate F1-scores. This parity demonstrates that *kernel switching* alone cannot overcome limitations in the input representation. Support Vector Machines excel when provided with informative, well-engineered features; changing the kernel only re-maps the same features into different spaces without adding new discriminative power. Consequently, our pipeline in its current form **cannot yet be deployed in real-life applications**, as the overall and per-class performance falls below practical thresholds for robust car model recognition.

Key Findings

- **Feature quality bottleneck:** All three kernels yield similar results, indicating that the SIFT+BoVW histograms lack sufficient separability for under-represented or visually similar classes.
- **Class imbalance effects:** Minority classes (Mitsubishi models) collapse entirely, with zero recall, underscoring the need for balanced training data.
- **Hyperparameter impact:** Limited tuning of the regularisation parameter C and codebook size further constrains performance gains.

Future Work

To elevate accuracy well beyond the current plateau[10], we recommend the following research directions:

1. **Advanced feature extraction:** Explore deep learning–based embeddings (e.g., pre-trained CNNs such as ResNet or MobileNet) or hybrid approaches combining handcrafted and learned features.
2. **Data augmentation and balancing:** Generate synthetic samples for under-represented classes via image augmentation (rotation, color jitter) or generative models (GANs).
3. **Comprehensive hyperparameter optimisation:** Perform a grid or Bayesian search over C, kernel-specific parameters (γ for RBF), and codebook sizes to fully exploit SVM capacity.
4. **Ensemble and meta-learning methods:** Combine multiple classifiers or feature sets to improve robustness and generalisation.
5. **Real-world evaluation:** Validate the improved pipeline on larger, more diverse datasets and in real-time scenarios (e.g., live video streams, varying lighting conditions).

By addressing the feature representation at its core and adopting richer, data-driven methods, future iterations of this project can push beyond the current limitations and approach accuracies suitable for real-life deployment.

VII. REFERENCES

[1] https://www.kaggle.com/datasets/abhishektyagi001/vehicle-make-model-recognition-dataset-vmmrdb/data

[2] https://www.youtube.com/watch?v=4AvTMVD9ig0.

[3] https://www.youtube.com/watch?v=a4cFONdc6nc

[4] https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[5] https://dl.acm.org/doi/abs/10.1145/3107614

[6] Car Images classification using CNN https://www.kaggle.com/code/kshitij192/car-images-classification-using-cnn

[7] https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

[8] https://www.pinecone.io/learn/series/image-search/bag-of-visual-words/

[9] https://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Algorithm

[10] https://github.com/kamwoh/Car-Model-Classification

[11] https://github.com/nghia182004/com--vision.git

Thank you, Professor Dung, for taking the time to review our report.