Scene Classification using Spatial Envelope (GIST) features

Principles of Biological Vision

- Marri Bharadwaj (B21CS045)

Introduction

The objective of this project is to classify landscape images into predefined categories—such as *coast*, *desert*, *forest*, etc—by leveraging holistic image features for context recognition. Specifically, *GIST descriptors*, which capture spatial layout and structure, are used to represent the scenes. Using these features, I aim to build and train a *Bayesian classifier* to differentiate among various landscape types. The classifier's performance will be rigorously evaluated on both training and test datasets to assess its accuracy, generalisation capabilities, and feature effectiveness in real-world scene classification scenarios.

Data Preprocessing and Exploration

Dataset Loading & Splitting

I defined paths for training, validation, and testing data, and then used *loading_images_and_categories* to read images and labels for each split. Each image was converted to greyscale for standardisation.

```
Unique Categories are ['Mountain', 'Coast', 'Desert', 'Forest', 'Glacier']
Number of Training images: 10000
Number of Validation images: 1500
Number of Testing images: 500
```

Sample Image Display

Then, I displayed sample images from each category in grayscale, allowing visual verification of image categories and quality as follows-

Mountain

Coast

Desert

Forest

Glacier

Label Encoding

The *label_encoding* function converts categorical labels into numerical form, using *LabelEncoder*.

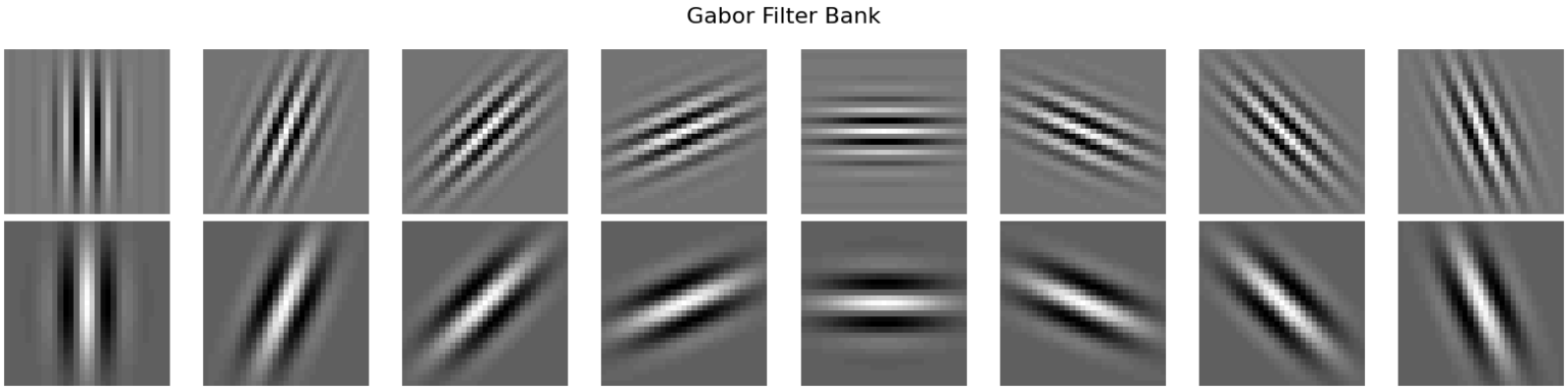
```
Label Classes:  ['Coast' 'Desert' 'Forest' 'Glacier' 'Mountain']
Encoded Labels:  [4 4 4 ... 3 3 3]
```

GIST Feature Extraction

Feature Extraction using GIST

I defined a class for *GIST* feature extraction. Some of the functions included in the class are:

- *resize_image*: Resizes each image to a specified dimension *(128x128)*, standardising the input size for consistent feature extraction.
- *_create_gabor_filters*: Generates a set of *Gabor filters* across different scales and orientations, enabling the capture of edge and texture information at multiple resolutions and angles.



- *extract_features_from_dataset*: Applies the feature extraction process to an entire dataset, collecting *GIST* features for each image into a feature matrix, preparing it for further processing or clustering.
- *get_gist_features*: Divides each image into a grid and applies the *Gabor filters* to calculate *GIST* features based on the mean values in each grid cell. This captures spatial layout and texture details in a structured feature array.

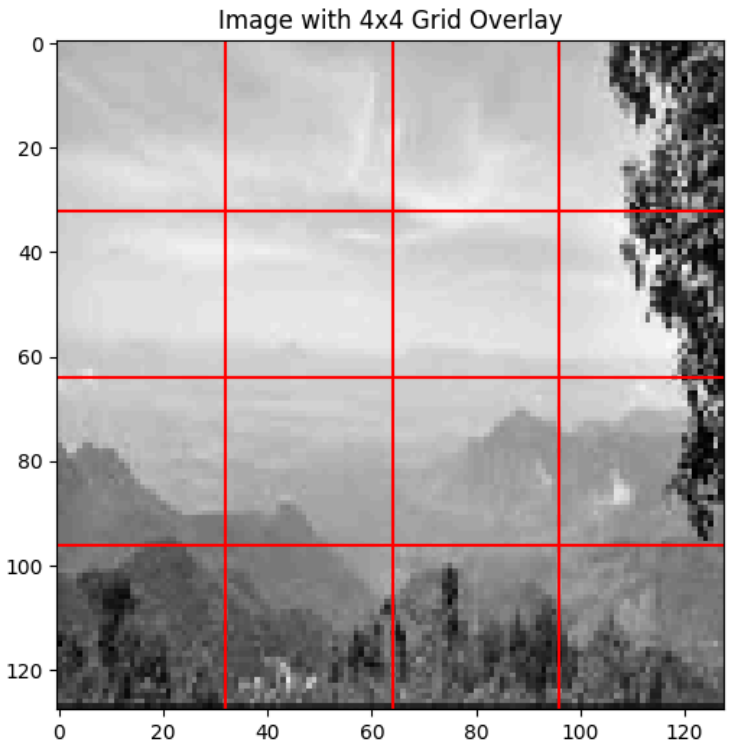
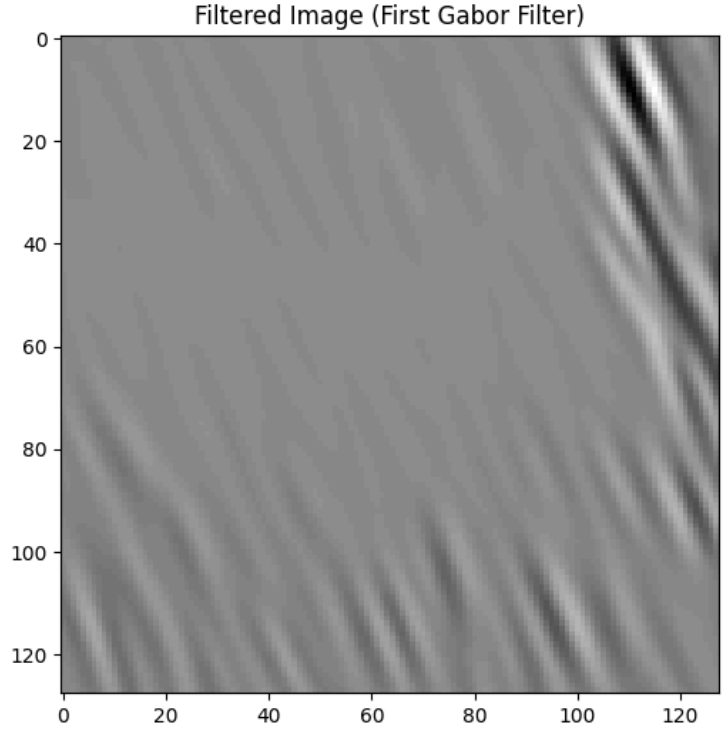
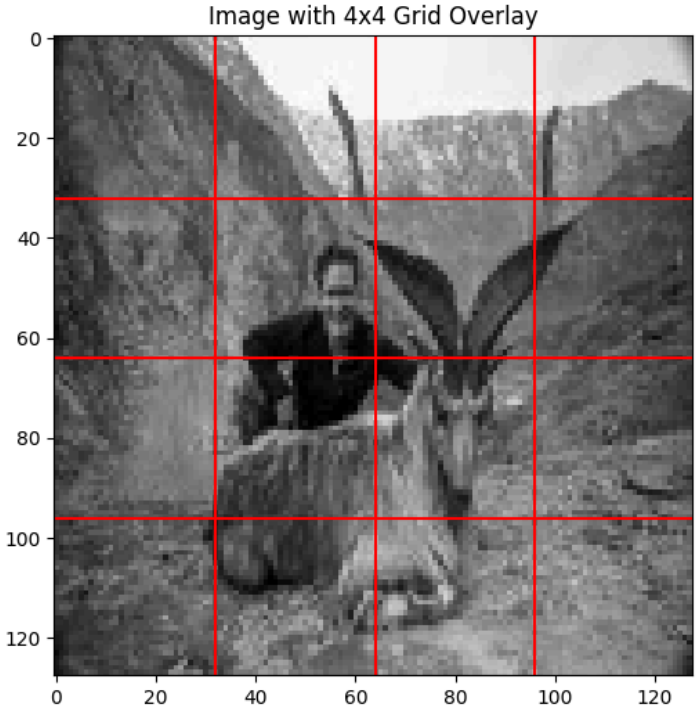
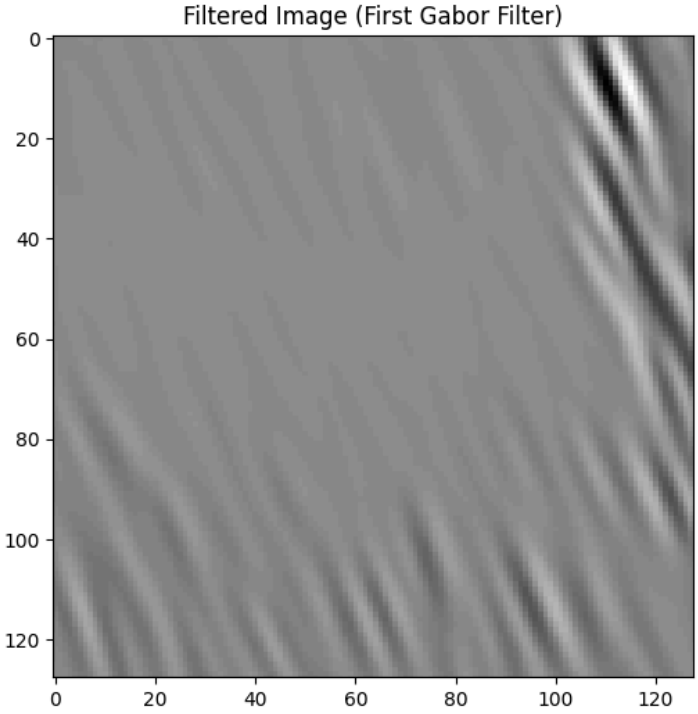


Image with 4x4 Grid Overlay
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Filtered Image (First Gabor Filter)
0
20
40
60
80
100
120
0
20
40
60
80
100
120

Feature Extraction on Test dataset



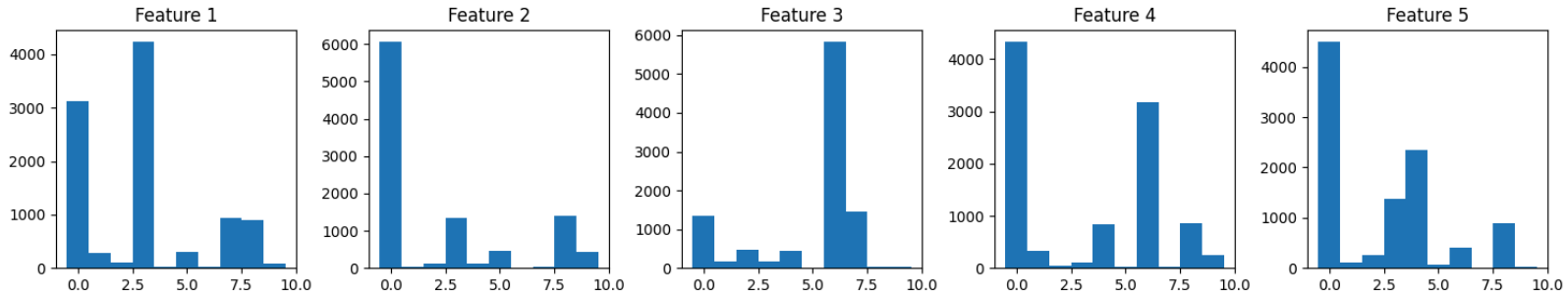


Feature Quantisation using K-Means Clustering

The *quantise_gist_features* function applies *K-Means Clustering* to the extracted *GIST* features for each feature dimension, quantizing them into a defined number of clusters (*num_clusters*). This reduces feature complexity by mapping continuous feature values to discrete states, enhancing model interpretability and reducing memory usage. I initially took ten clusters.

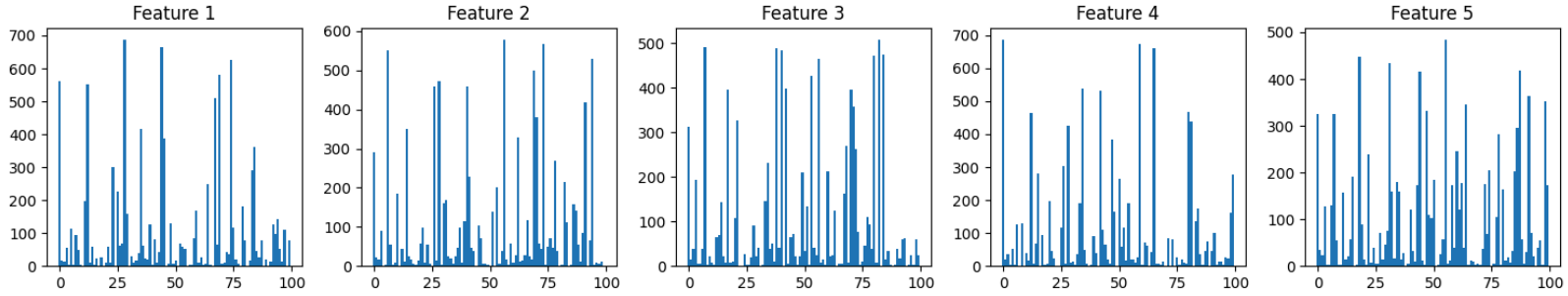
```
Quantisation completed!
Original feature range: [-242.331, 610.138]
Quantised feature range: [0, 9]
Number of quantisation levels per feature is 10
```

The *display_quantisation_distribution* function visualises the quantised state distribution across selected features, using histograms to show frequency for each quantisation level. By plotting the first *num_features* (default 5), it highlights how well clusters are represented, aiding in assessing balance and identifying any over- or under-representation among quantised states.



Increasing to 100 clusters

```
Quantisation completed!
Original feature range: [-242.331, 610.138]
Quantised feature range: [0, 99]
Number of quantisation levels per feature is 100
```



Bayesian Network Classifier

I defined a class named *BayesianNetworkClassifier*. Some of the functions defined in it are:

- *fit*: Trains the classifier on the training data, *X_train* and *y_train*
- *_calculate_prior*: Computes the prior probability of each class as the ratio of class occurrences to total samples in *y_train*.
- *_calculate_likelihoods*: Calculates likelihoods for each feature value within a class. Applies Laplace smoothing to handle unseen values, storing these as probability mappings for each feature in each class.
- *predict*: Predicts the class labels for the test data *X_test*. For each sample, it calculates the log-posterior for each class based on prior and likelihoods, selecting the class with the highest posterior probability.
- *evaluate*: Evaluates model performance on *X_test* and *y_test*. It calculates accuracy, generates a classification report, and plots a confusion matrix. Allows specifying class names for more interpretable output.
- *plot_confusion_matrix*: Plots the confusion matrix, with an option to normalise values. Displays the matrix with coloured cells and labels, enhancing visualisation of classification performance across classes.

At last, I applied the *Bayesian Classifier* on the quantised feature images

Testing and Evaluation

Evaluation Metrics

The *evaluate* function uses three main performance metrics: *Accuracy*, *Precision*, *Recall*, and *F1-score* and *Confusion Matrix*.

- *Accuracy* gives an overall correctness rate of predictions.
- *Precision*, *Recall*, and *F1-score* provide a deeper breakdown of performance per class, highlighting areas where the classifier is effective or struggling.
- *Confusion Matrix* visualises true v/s predicted classifications, showing misclassification patterns.

Quantising for 10 clusters

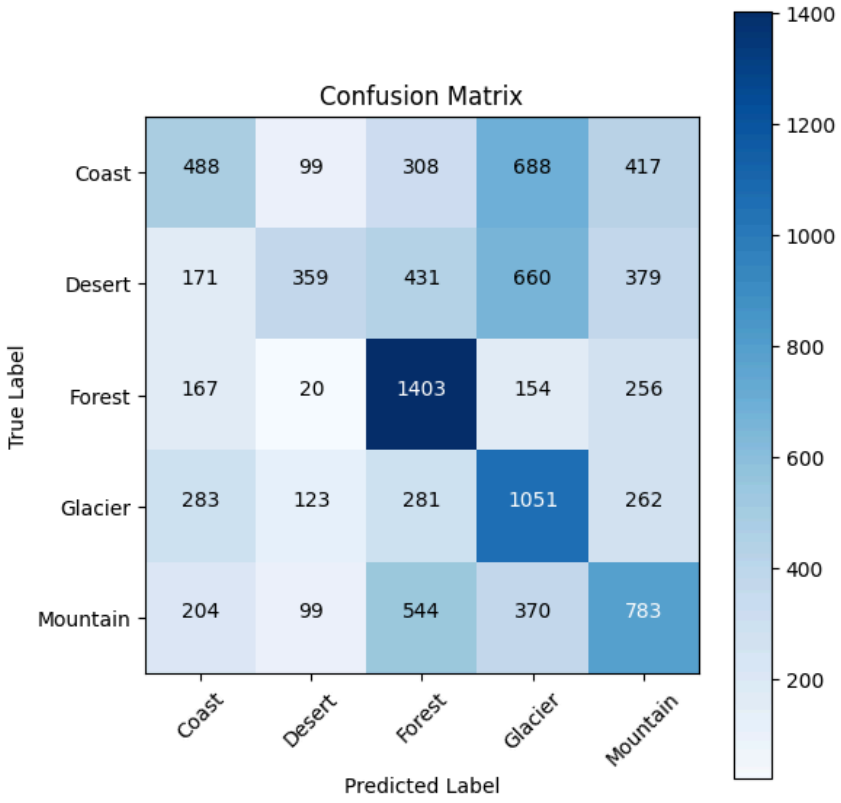
- Evaluating on Training set

METRICS

Accuracy: 40.84%

	precision	recall	f1-score	support
Coast	0.37	0.24	0.29	2000
Desert	0.51	0.18	0.27	2000
Forest	0.47	0.70	0.56	2000
Glacier	0.36	0.53	0.43	2000
Mountain	0.37	0.39	0.38	2000
accuracy			0.41	10000
macro avg	0.42	0.41	0.39	10000
weighted avg	0.42	0.41	0.39	10000

CONFUSION MATRIX



Quantising for 100 clusters

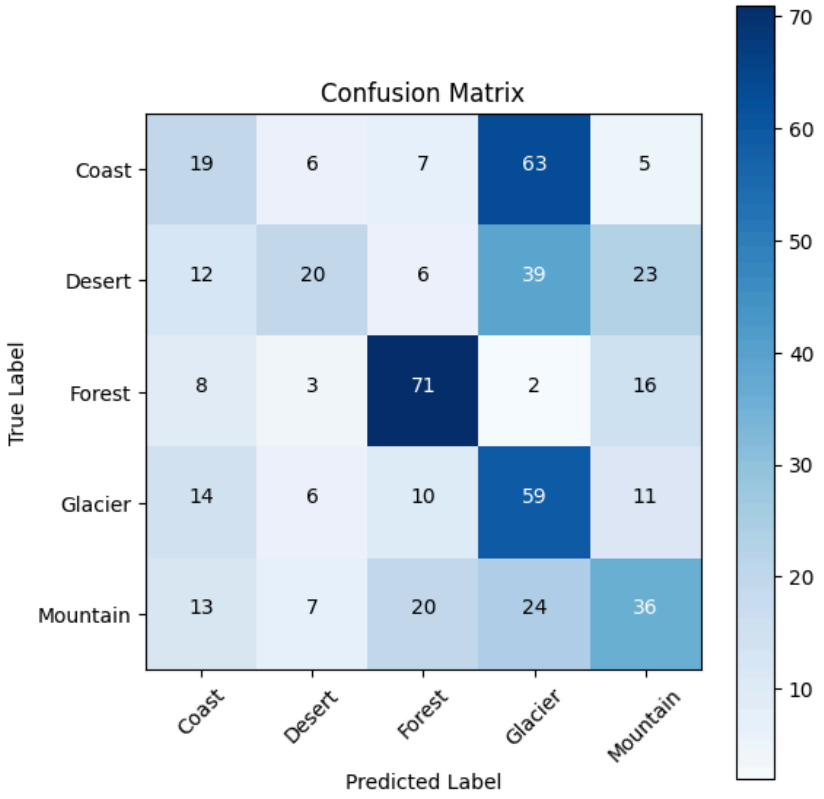
- Evaluating on Training set

METRICS

Accuracy: 41.00%

	precision	recall	f1-score	support
Coast	0.29	0.19	0.23	100
Desert	0.48	0.20	0.28	100
Forest	0.62	0.71	0.66	100
Glacier	0.32	0.59	0.41	100
Mountain	0.40	0.36	0.38	100
accuracy			0.41	500
macro avg	0.42	0.41	0.39	500
weighted avg	0.42	0.41	0.39	500

CONFUSION MATRIX



Comparison and Analysis of Results

Analysing and Comparing the results obtained on train and test sets:

A. 10 clusters

1. Training Set Performance

a. *Training Accuracy* is 40.84%, indicating moderate effectiveness in recognising patterns within the training data.

b. Class Performance:

- *Forest* has the highest *recall* (0.70) and *F1-score* (0.56), showing the model's ability to identify this class accurately.
- *Desert* and *coast* classes have lower *recall* values (0.18 and 0.24, respectively), indicating frequent misclassification.

c. *Confusion Matrix*: *Forest* is mostly correctly classified, while *coast* and *desert* are commonly misclassified as *glacier* and *mountain*, indicating overlaps in feature representations among these classes.

2. Testing Set Performance

a. *Testing Accuracy* is 41%, similar to training accuracy, showing stable but limited generalisation.

b. Class Performance:

- *Forest* again performs best with the highest *recall* (0.71) and *F1-score* (0.66), suggesting the model's learned patterns for this class generalise well.
- *Desert* and *coast* continue to have low *recall* values (0.20 and 0.19), indicating persistent difficulty in identifying these classes accurately.

c. *Confusion Matrix*: Similar misclassification patterns as the training set, with *coast* and *desert* frequently misclassified as *glacier* and *mountain*, suggesting a need for improved feature separation.

B. 100 clusters

1. Training Set Performance

a. *Training Accuracy* is 57.97%, improved over previous trials, indicating better fit on training data with higher quantisation.

b. Class Performance:

- *Forest* has the highest *recall* (0.75) and *F1-score* (0.61), showing strong recognition of this class.
- *Desert* and *mountain* have lower *recall* (0.42 and 0.52, respectively), indicating more difficulty in identifying these classes accurately.
- Overall, *macro* and *weighted F1-scores* at 0.58 reflect balanced improvement in performance.

c. *Confusion Matrix*: *Forest* is still the best-classified class, while *coast* and *desert* are commonly misclassified as *glacier* and *mountain*, though less frequently than before, suggesting that finer quantisation captures more unique features per class.

2. *Testing Set Performance*

a. *Testing Accuracy* is 46.2%, lower than training accuracy, showing overfitting with the finer quantisation level.

b. Class Performance:

- *Forest* continues to have the highest *recall* (0.75) and *F1-score* (0.70), showing it generalises better for this class.
- *Desert* and *coast* show lower *precision* and *recall*, indicating a tendency for misclassification.
- *Macro* and *weighted averages of F1-scores* (0.46) are lower than the training set, highlighting reduced generalisation across classes.

c. *Confusion Matrix*: Misclassification patterns are similar to the training set, with *coast* and *desert* often predicted as *glacier* and

mountain, suggesting persistent feature overlap despite higher quantisation.

Conclusion

- The model shows improved *accuracy* on the training set with 100 clusters, but this comes with increased overfitting, as shown by the drop in test accuracy.
- Class-level performance gains for *forest* are evident in both sets, but other classes, particularly *coast* and *desert*, still experience notable misclassifications.
- Higher quantisation aids in capturing class-specific features, though further tuning may be needed to balance fit and generalisation.