## **Image Classification Using Handcrafted Features**

### **1. Image Preprocessing Steps**

* **Image Loading**: Images are loaded from the dataset directory.
* **Resizing**: Each image is resized to a fixed size of 128x128 pixels to ensure uniformity.
* **Grayscale Conversion**: The resized images are converted to grayscale to reduce complexity and focus on intensity patterns.
* **Normalization**: The pixel values are normalized to the range [0, 1] by dividing by 255.0 to standardize the input.

### **2. Feature Selection**

* **Histogram of Oriented Gradients (HOG)**: Captures edge and gradient information which is crucial for identifying shapes and patterns.
* **Gabor Filters**: These filters are used to detect texture information at different scales and orientations.
* **Local Binary Pattern (LBP)**: Captures local texture information by comparing each pixel with its neighbors.
* **Scale-Invariant Feature Transform (SIFT)**: Extracts keypoints and descriptors that are invariant to scale and rotation, making them useful for identifying distinct features.

These features collectively provide a comprehensive representation of the images, improving the model's ability to distinguish between different classes.

### **3. Evaluation of the Trained Models**

The model is evaluated using standard metrics:

* **Accuracy**: Measures the overall correctness of the model.
* **Classification Report**: Provides detailed metrics such as precision, recall, and F1-score for each class.
* **Confusion Matrix**: Shows the true positives, true negatives, false positives, and false negatives for each class.

python

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

Accuracy: 0.7550047664442326

Classification Report:

precision recall f1-score support

Building 0.68 0.51 0.58 92

Forest 0.83 0.99 0.90 526

Glacier 0.65 0.55 0.59 106

Mountains 0.60 0.60 0.60 112

Sea 0.62 0.54 0.57 108

Streets 0.77 0.41 0.53 105

accuracy 0.76 1049

macro avg 0.69 0.60 0.63 1049

weighted avg 0.74 0.76 0.74 1049

Confusion Matrix:

[[ 47 24 3 5 2 11]

[ 0 519 0 4 2 1]

[ 1 20 58 13 13 1]

[ 2 14 11 67 18 0]

[ 3 13 12 22 58 0]

[ 16 39 5 1 1 43]]

### **4. Development of a Flask Application with Image Upload Functionality for Classification**

The Flask application allows users to upload an image, which is then preprocessed and classified using the trained model. The application includes endpoints for rendering the upload page and handling file uploads:

python

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

# File upload and prediction logic

### 

### **5. Setting Up a Flask Application for Local Image Classification**

1. **Install Dependencies**: Ensure you have all necessary libraries installed (flask, opencv-python, numpy, scikit-image, scikit-learn, joblib).
2. **Run the Flask App**: python app.py
3. **Access the Application**: Open a web browser and go to http://127.0.0.1:5000/.

### **6. Enhancement Scope to Improve the Performance of the Model**

* **Hyperparameter Tuning**: Experiment with different hyperparameters for the RandomForestClassifier and PCA.
* **Feature Engineering**: Explore additional handcrafted features or combinations thereof.
* **Automated Feature Extraction**: While this assignment avoids deep learning, using pre-trained CNN models for feature extraction (as a future enhancement) can significantly improve performance.

### **Automating Feature Extraction Process**

To automate feature extraction, you could integrate the feature extraction functions into a pipeline that processes all images and extracts features in one step:

python

def extract\_features(images):

return np.array([extract\_combined\_features(img) for img in images])

This function can be called directly after loading the images to streamline the preprocessing and feature extraction workflow.