Project Report

Objective

This project aims to build an image classification system that can categorize images of **sea animals** into multiple classes using **traditional machine learning techniques**. Instead of deep learning or transfer learning, we extract **handcrafted features** from the images and train models like **Support Vector Machines (SVM)** and **Random Forests (RF)** to perform classification.

Dataset Description

- **Name**: Sea Animals Image Dataset (from Kaggle)
- **Format**: ZIP archive containing image folders, each representing a class of sea animal (e.g., fish, shark, crab, etc.)
- **Size**: Large (~several thousand images), with varying image resolutions

Tools & Technologies Used

- **Google Colab** (for training environment)
- **Python**
- **Libraries**:
  - OpenCV – image processing
  - scikit-image – LBP feature extraction
  - scikit-learn – ML models and evaluation
  - joblib – model saving/loading
  - matplotlib, seaborn – visualization
  - tqdm – progress bar
  - os, glob, numpy – file and data handling

Pipeline Workflow

1. Google Drive Integration

- Mounted Google Drive to Colab to upload and access large ZIP dataset.
- Extracted dataset using Python's zipfile module.

2. Data Preprocessing

- Images were resized to a fixed dimension (**128x128**) for uniformity.
- File paths were indexed and labeled using folder names (class labels).

3. Handcrafted Feature Extraction

For each image, three sets of features were extracted:

- **Color Histogram**:
    - Captures the distribution of colors (HSV space).
    - 128-bin histogram used for reduced feature size.
- **Local Binary Patterns (LBP)**:
    - Texture descriptor capturing local patterns in grayscale.
    - Radius = 1, Points = 8.
- **Hu Moments**:
    - Shape descriptor based on image moments (invariant to rotation/scale).

All features were **concatenated** into a single feature vector per image.

4. Data Preparation

- Feature vectors and labels were encoded using LabelEncoder.
- Data was split into **training (80%)** and **testing (20%)** using stratified sampling.

Model Training & Evaluation

Models Used:

1. **Support Vector Machine (SVM)** with RBF kernel
2. **Random Forest** (n_estimators = 50)

(Gradient Boosting was excluded due to compute limitations)

Evaluation Metrics:

Each model was evaluated using:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-score**

- **Confusion Matrix**

Confusion matrices were visualized using heatmaps to understand per-class performance.

Model Saving

The best-performing model (Random Forest) was saved using joblib:

joblib.dump(models["Random Forest"], "sea_animals_rf_model.pkl")

This allows the model to be loaded and used later for prediction without retraining.

Key Learnings

- Traditional ML combined with handcrafted features can still yield strong performance for image classification.
- Texture (LBP), color (histogram), and shape (Hu Moments) features together create a powerful combination.
- Optimization and preprocessing decisions can significantly affect model training time and accuracy.
- Real-world constraints (hardware limits) sometimes guide architectural decisions — like skipping Gradient Boosting.

Conclusion

This project successfully demonstrates a complete **image classification pipeline** without deep learning. By using a mix of classical image processing techniques and robust ML models, we achieved high accuracy while keeping the system lightweight and interpretable.