A Binary Classification Using Handcrafted Features and ML Classifiers

1 Introduction

This project focuses on binary classification for mask detection using handcrafted feature extraction techniques combined with machine learning models. The goal is to classify images into two categories: "with_mask" and "without_mask." We employ a robust feature extraction pipeline and evaluate multiple classifiers, including SVM, MLP, XGBoost, and RandomForest.

Workflow

- Extract features from images using HOG, Color Histograms, LBP, and Edge Histograms.
- Train multiple machine learning models for classification.
- Use Google Colab for accelerated model training.
- Perform final model evaluation on a local machine.

Directory Structure

```
C:.
   A_README.MD
   Colab_USAGE_ML.ipynb
   feature_extraction.py
   main.ipynb

enhanced_features
       X.npy
       y.npy

saved_models
       accuracy_log.txt
       MLP.pth
       MLP_report.txt
       RandomForest.pkl
```

```
    RandomForest_report.txt
    SVM.pkl
    SVM_report.txt
    XGBoost.pkl
    XGBoost_report.txt

plots
        MLP_confusion_matrix.png
        RandomForest_confusion_matrix.png
        SVM_confusion_matrix.png
        XGBoost_confusion_matrix.png
```

2 Feature Extraction

We extract four types of handcrafted features from images to capture different aspects of the data:

2.1 Histogram of Oriented Gradients (HOG)

HOG captures the structure and texture of an image by computing gradient orientations in localized regions.

2.2 Color Histogram

Color histograms capture the distribution of colors in an image, making them useful for distinguishing different object categories.

2.3 Local Binary Patterns (LBP)

LBP is a texture descriptor that encodes local patterns in an image based on pixel intensity differences.

2.4 Edge Histogram

Edge histograms capture the distribution of edge intensities in an image.

3 Google Colab for Accelerated Model Training

Since training multiple machine learning models can be computationally expensive, we utilized Google Colab to leverage free GPU resources.

3.1 Colab Workflow

- Load extracted features from `enhanced_features/X.npy` and `enhanced_features/y.npy`.
- Train four models: SVM, MLP, XGBoost, and RandomForest.

- Save trained models in `saved_models/` for later evaluation.

4 Model Evaluation on Local Machine

Once the models were trained in Colab, they were downloaded and evaluated in `main.ipynb`.

4.1 Final Model Results

Model	Accuracy
SVM	93.87%
MLP	93.25%
XGBoost	92.64%
RandomForest	90.06%

Table 1: Final Model Accuracy

5 Model Reports and Confusion Matrices

Each trained model has a detailed classification report and a corresponding confusion matrix.

5.1 SVM

	precision	recall	f1-score	support
0	0.96	0.93	0.94	446
1	0.91	0.95	0.93	369
accuracy			0.94	815
macro avg	0.94	0.94	0.94	815
weighted avg	0.94	0.94	0.94	815

5.2 MLP

	precision	recall	f1-score	support
0	0.94	0.94	0.94	446
1	0.92	0.93	0.93	369
accuracy			0.93	815
macro avg	0.93	0.93	0.93	815
weighted avg	0.93	0.93	0.93	815

6 Conclusion

Among the four models evaluated, SVM achieved the highest accuracy (93.87%), making it the best-performing classifier for this binary classification task. MLP (93.25%) and XGBoost (92.64%) followed closely, while RandomForest (90.06%) had the lowest accuracy.

The results indicate that SVM effectively leveraged the handcrafted features (HOG, LBP, Color Histograms, and Edge Histograms) to achieve superior classification performance. The slight drop in MLP and XGBoost performance suggests that while deep learning and boosting methods are powerful, the handcrafted features might be more naturally suited for linear separability, which benefits SVM.

In summary, SVM is the most optimal model for this feature extraction approach, offering the best balance between accuracy and computational efficiency.