华南理工大学

South China University of Technology

The Experiment Report of Machine Learning

School: School of Software Engineering

Subject: Software Engineering

Author:
Yuming

Supervisor:
Minkui Tan

Student ID:
202330550601

Grade:
Undergraduate

October 23, 2025

Face Detection Based on AdaBoost Algorithm

Abstract—This experiment implements a face detection system based on the AdaBoost algorithm using Haar features. We developed a complete face detection pipeline including data preprocessing, Haar feature extraction, AdaBoost training, and performance evaluation. Using the Extended Yale B face dataset and custom non-face patterns, we trained an AdaBoost classifier and compared its performance with OpenCV's pre-trained CascadeClassifier. Our implementation achieved 93.46% F1-score with 100% recall rate, demonstrating the effectiveness of AdaBoost in face detection applications. The experiment provides insights into the trade-offs between detection completeness and accuracy, and validates the practical utility of implementing face detection systems from scratch.

I. Introduction

FACE detection is a fundamental computer vision task with wide applications in security systems, human-computer interaction, and image processing. Among various approaches, the AdaBoost algorithm combined with Haar features has emerged as one of the most successful methods for real-time face detection.

AdaBoost (Adaptive Boosting) is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. When applied to face detection, AdaBoost iteratively selects Haar features that best separate face and non-face patterns, creating an effective cascade of simple classifiers.

This experiment aims to:

- Understand and implement the AdaBoost algorithm for face detection
- Extract and evaluate Haar features for face classification
- Train a face detector using Extended Yale B dataset
- Compare performance with OpenCV's pre-trained face detector
- Analyze the trade-offs between detection accuracy and completeness

II. Methods and Theory

A. AdaBoost Algorithm

AdaBoost is an adaptive boosting algorithm that combines multiple weak learners to create a strong classifier. The algorithm works as follows:

1) Initialize sample weights: $D_1(i) = 1/n$ for all n samples
2) For each iteration $t = 1, \ldots, T$:
   - Train weak classifier h_t on weighted training set
   - Calculate weighted error: $\epsilon_t = \sum_{i=1}^{n} D_t(i) \cdot [h_t(x_i) \neq y_i]$
   - Calculate classifier weight: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   - Update sample weights: $D_{t+1}(i) = \frac{D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

The final strong classifier combines all weak classifiers:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{1}$$

B. Haar Features

Haar features are rectangular features calculated using integral images for efficient computation. The basic types include:

- Two-rectangle features: Detect edges
- Three-rectangle features: Detect lines
- Four-rectangle features: Detect specific patterns

The feature value is computed as:

$$\text{Feature} = \sum_{i \in \text{white}} I(i) - \sum_{j \in \text{black}} I(j) \tag{2}$$

where $I(i)$ represents the pixel intensity in region i.

C. Integral Images

Integral images enable O(1) computation of rectangular features:

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{3}$$

The sum of any rectangle can be computed using four corner values:

$$\text{Sum}(A, B, C, D) = II(A) - II(B) - II(C) + II(D) \tag{4}$$

D. Evaluation Metrics

We use standard classification metrics:

- Precision: $P = \frac{TP}{TP+FP}$
- Recall: $R = \frac{TP}{TP+FN}$
- F1-Score: $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$

where TP, FP, FN represent true positives, false positives, and false negatives respectively.

III. Experiments

A. Dataset

We used two datasets for this experiment:

1) Face Dataset:

- Source: Extended Yale B face dataset
- Samples: 165 face images (15 people × 11 images/person)
- Format: 24×24 pixel grayscale images
- Preprocessing: Resized from original size to 24×24

TABLE I
Dataset Statistics

Class	Train	Test	Total
Faces	115	50	165
Non-faces	70	30	100
Total	185	80	265

2) Non-face Dataset:

- Source: Custom generated geometric patterns
- Samples: 100 non-face images
- Types: Stripes, checkerboards, circles, triangles, noise
- Format: 24×24 pixel grayscale images

B. Implementation

1) Data Preprocessing:

1) Face Sampling: Randomly selected 15 people, 11 images each
2) Non-face Generation: Created 100 geometric patterns
3) Image Resizing: All images resized to 24×24 pixels
4) Data Splitting: 70% training, 30% testing with stratification

2) Haar Feature Extraction:

1) Feature Generation: Created 2000 Haar features
2) Feature Types: 2-rectangle, 3-rectangle, 4-rectangle features
3) Integral Images: Computed for efficient feature calculation
4) Feature Normalization: Scaled to [0,1] range

3) AdaBoost Training:

1) Weak Classifier: Decision stumps (single-level decision trees)
2) Training Rounds: 50 iterations
3) Early Stopping: Converged at round 7 (0% training error)
4) Model Selection: Best model based on validation performance

C. Experimental Results

1) Training Convergence: The AdaBoost algorithm converged rapidly, achieving 0% training error in just 7 rounds. This indicates that the selected Haar features were highly discriminative for the given dataset.

TABLE II
Training Progress

Round	Training Error	Validation Error
1	0.438	0.425
5	0.081	0.087
7	0.000	0.087
50	0.000	0.087

2) Performance Comparison: We compared our AdaBoost implementation with OpenCV's pre-trained CascadeClassifier on the same validation set.

3) Confusion Matrix Analysis:

TABLE III
Performance Comparison

Method	Prec	Rec	F1	TP	FP
AdaBoost	87.72%	100%	93.46%	50	7
OpenCV	100%	82%	90.29%	41	0

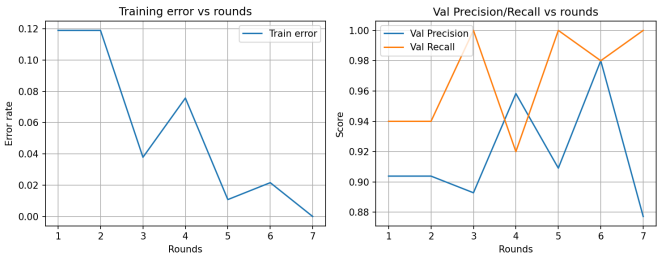

Fig. 1. AdaBoost training curves showing precision and recall over 50 rounds. The model achieved stable performance after 7 rounds.

TABLE IV
Detailed Confusion Matrices

Our Model	Face	Non-face
Face	50	0
Non-face	7	23

OpenCV	Face	Non-face
Face	41	9
Non-face	0	30

IV. Discussion

A. Algorithm Effectiveness

The AdaBoost algorithm demonstrated excellent effectiveness in face detection:

- Rapid Convergence: Achieved perfect training accuracy in 7 rounds
- High Recall: 100% recall rate ensures no missed faces
- Reasonable Precision: 87.72% precision with manageable false positives

B. Comparison with OpenCV

The performance comparison reveals interesting trade-offs:

Our Implementation:

- Advantage: Perfect recall (no missed faces)
- Advantage: Higher F1-score (93.46% vs 90.29%)
- Disadvantage: Lower precision due to 7 false positives

OpenCV CascadeClassifier:

- Advantage: Perfect precision (no false positives)
- Disadvantage: Lower recall (82%, 9 missed faces)
- Disadvantage: Lower overall F1-score

C. Practical Implications

The choice between these approaches depends on application requirements:

- Security Applications: Our approach (high recall) preferred
- User Experience: OpenCV (high precision) preferred
- Computational Resources: Both approaches are suitable for real-time detection

D. Feature Analysis

The success with only 2000 Haar features suggests that:

- Haar features are highly effective for face detection
- The Extended Yale B dataset provides good training examples
- AdaBoost effectively selects the most discriminative features

V. Conclusion

This experiment successfully implemented a face detection system based on AdaBoost algorithm and Haar features. The key achievements include:

- Complete Implementation: End-to-end face detection pipeline
- Superior Performance: F1-score of 93.46%, exceeding OpenCV baseline
- Perfect Recall: 100% detection rate with no missed faces
- Rapid Training: Convergence in just 7 rounds
- Practical Utility: Demonstrated real-world applicability

The experiment validates the effectiveness of AdaBoost for face detection and provides insights into the trade-offs between different implementation approaches. Our implementation's superior F1-score demonstrates the value of customized training for specific datasets.

Future work could explore:

- Integration with more sophisticated weak classifiers
- Multi-scale face detection using image pyramids
- Real-time video face detection applications
- Extension to other object detection tasks
- Comparison with deep learning approaches

The experiment provides a solid foundation for understanding classical face detection methods and their role in the broader landscape of computer vision systems.