Face Detection System Using HOG, LBP and SVM

Hoza Violeta Maria
Technical University of Cluj-Napoca

***Abstract*—This paper presents a robust face detection system based on classical computer vision techniques, combining Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) for feature extraction with Support Vector Machine (SVM) classification. The system addresses the challenge of detecting human faces in color images under varying conditions including lighting changes, noise, distance variations, and partial occlusions. The proposed solution achieves competitive performance through careful feature engineering, multi-scale detection, and hard negative mining strategies. Experimental results on a diverse dataset demonstrate the effectiveness of the classical approach, achieving precision of 69.94%, recall of 58.46%, and F1-score of 63.69% on 100 test images. The system includes a user-friendly GUI for training, detection, and evaluation, making it accessible for practical applications.**

I. Introduction

A. Context

Face detection is a fundamental problem in computer vision with applications spanning security systems, human-computer interaction, photography, and social media platforms. The ability to automatically locate and identify human faces in digital images has become increasingly important as visual data continues to grow exponentially. Despite advances in deep learning approaches, classical computer vision methods remain relevant due to their interpretability, lower computational requirements, and effectiveness in resource-constrained environments.

B. Problem Statement

The challenge addressed in this project is the robust detection of human faces in color images containing various objects. Specifically, the system must:

- Detect faces invariant to lighting conditions
- Handle images with noise and varying quality
- Detect faces at different distances (scales)
- Work with partial face views and occlusions
- Process images containing single or multiple faces
- Mark detected faces with bounding rectangles

Traditional approaches face several challenges including computational complexity, false positives in cluttered backgrounds, and sensitivity to pose variations. This project addresses these challenges through a combination of robust feature extraction and machine learning classification.

C. Proposed Objectives

The main objectives of this project are:

1) Implement a robust feature extraction pipeline combining HOG and LBP descriptors
2) Train an effective SVM classifier for face/non-face discrimination
3) Develop a multi-scale sliding window detection framework
4) Implement Non-Maximum Suppression (NMS) to eliminate duplicate detections
5) Create comprehensive evaluation metrics including precision, recall, and F1-score
6) Design an intuitive graphical user interface for system interaction
7) Validate the system on a dataset with ground truth annotations

II. Theoretical Background

A. Bibliographical Study

Face detection is a fundamental task in computer vision, often serving as a precursor for recognition, tracking, or expression analysis. Unlike controlled settings, real-world images contain diverse objects, lighting conditions, and occlusions, all of which complicate the detection process. Historically, algorithms such as Viola–Jones laid the groundwork for real-time detection. Later methods, such as HOG + SVM, improved robustness to scale and orientation. More recently, deep learning approaches—particularly CNN-based architectures—have surpassed classical methods in accuracy and generalization. Given the diversity of input images (color images with various single objects), a detection method must extract meaningful features and handle ambiguity efficiently, even when the presence of a face is uncertain.

B. Overview of Traditional Methods

1) Viola-Jones Algorithm: The **Viola-Jones** algorithm (2001) is a foundational framework for real-time face detection, particularly effective for frontal faces. The key components of the algorithm are:

1) **Haar-like Features**: The algorithm uses simple rectangular features resembling Haar wavelets to capture the presence of edges, lines, and other texture elements of faces. These features are computed efficiently using an integral image representation, which speeds up calculation.
2) **AdaBoost Training:** Among thousands of Haar features, AdaBoost selects a small number of critical features and constructs a strong classifier as a weighted combination of weak classifiers, each corresponding to one Haar feature. This reduces the dimensionality and selects the most discriminative features.

3) **Cascade of Classifiers:** Instead of applying a complex classifier on every image sub-window, Viola-Jones employs a cascade structure where simpler classifiers quickly discard negative regions. Only promising regions pass to later, more complex stages. This significantly speeds detection.
4) **Sliding Window:** The cascade classifier is applied to the image using a sliding window approach across multiple scales, enabling the detection of faces at various sizes and positions within the image.

2) Histogram of Oriented Gradients (HOG): HOG is a feature descriptor that captures the distribution of gradient orientations in localized regions of an image. Originally developed for pedestrian detection, HOG has proven effective for face detection due to its ability to capture edge and shape information while remaining robust to illumination changes. The key insight is that local object appearance and shape can be characterized by the distribution of local intensity gradients, even without precise knowledge of gradient positions.

3) Feature-based Methods : Feature-based methods detect faces by extracting and analyzing distinctive characteristics from images, rather than examining raw pixel values directly. These methods identify discriminative features that distinguish faces from non-faces, such as edges, textures, colors, and geometric patterns. The extracted features are then fed into classifiers to make detection decisions.

Common feature types are:

- **Color features:** Exploit skin color properties in various color spaces (RGB, HSV) to segment potential face regions based on skin tone distribution.
- **Texture features:** Capture local patterns using methods like Local Binary Patterns (LBP), which encode pixel relationships, or Gabor filters, which respond to specific orientations and frequencies.
- **Edge and gradient features:** Use edge detection (Canny, Sobel) or gradient-based descriptors like HOG to capture facial contours and structural information.
- **Geometrical features:** Identify and analyze spatial relationships between facial components (eyes, nose, mouth) based on their relative positions and proportions.

While feature-based methods are efficient and interpretable, they require manual feature design and may be less robust than deep learning approaches under extreme conditions.

LBP is a texture descriptor that encodes local spatial patterns by comparing each pixel with its neighbors. Its effectiveness stems from computational simplicity and invariance to monotonic gray-scale transformations. LBP captures complementary information to HOG: while HOG describes shape through gradients, LBP describes texture through local intensity patterns. The uniform pattern variant reduces dimensionality while maintaining discriminative power for face detection.

4) Template Matching: Template matching for face detection involves comparing segments of an input image to predefined face templates by sliding the template across the image and calculating a similarity measure (such as cross-correlation or sum of squared differences) at each position to find matching regions.

This approach does not require complex feature extraction or machine learning training and is straightforward to implement. It relies on pixel intensity comparisons between the candidate region and the template. Template matching works well when face appearance is consistent and under controlled conditions.

However, the method is sensitive to variations in scale, pose, lighting, partial occlusions, and facial expressions. The requirement to match predefined templates means it struggles with faces at different angles or partially visible. It is also computationally expensive if templates are large or many scales and rotations need to be tried.

In practice, template matching can be combined with other preprocessing steps such as color segmentation for skin detection or followed by more robust classifiers to enhance accuracy and speed.

C. Neural Network-based Methods

Neural network-based methods, particularly deep learning approaches using Convolutional Neural Networks (CNNs), represent the current state-of-the-art in face detection. These methods automatically learn hierarchical feature representations directly from raw image data, eliminating the need for manual feature engineering.

Deep neural networks learn to detect faces through multiple layers of processing. Lower layers capture basic patterns like edges and textures, middle layers identify facial components (eyes, nose, mouth), and higher layers recognize complete face structures and their variations.

Networks are trained on large labeled datasets (thousands to millions of images) using backpropagation. The model learns to minimize detection errors by adjusting millions of parameters. Data augmentation techniques (rotation, scaling, brightness changes) improve robustness to real-world variations.

Key architectures:

- **Region-based CNNs (R-CNN Family):** Methods like R-CNN, Fast R-CNN, and Faster R-CNN use region proposal networks to identify candidate face regions, then classify and refine bounding boxes through CNN processing.
- **Single-Shot Detectors**: **YOLO** (You Only Look Once) and **SSD** (Single Shot Detector) perform detection in a single pass through the network, achieving real-time performance by predicting bounding boxes and class probabilities directly.
- **Multi-Task Cascaded CNN (MTCNN):** Uses a cascade of three CNNs to progressively refine face detection and simultaneously perform facial landmark localization, balancing speed and accuracy.
- **RetinaNet & Feature Pyramid Networks:** Employ multi-scale feature pyramids to detect faces at different

sizes effectively, addressing the challenge of scale variation.

D. Support Vector Machines

SVMs are supervised learning models for classification that find the optimal separating hyperplane with maximum margin between classes. For linearly non-separable data, the kernel trick maps features to higher-dimensional spaces where separation becomes possible. The RBF (Radial Basis Function) kernel is particularly effective for face detection, as it can model complex decision boundaries between face and non-face patterns in the high-dimensional feature space.

III. Design and Implementation

A. Solution Overview

The proposed approach is a hybrid, feature-based face detection system built using classical computer vision methods. The design combines two complementary feature extraction techniques: the **Histogram of Oriented Gradients (HOG)**, which captures the overall shape and edge structure of a face, and the **Local Binary Patterns (LBP)**, which describe fine texture variations. These features are concatenated to form a robust feature vector that represents each image patch.

A **Support Vector Machine (SVM)** classifier is trained using these feature vectors, with positive samples taken from known face regions and negative samples from non-face areas. During detection, the trained SVM evaluates sliding windows at multiple image scales to determine whether each region contains a face. Finally, overlapping detections are refined using Non-Maximum Suppression (NMS) to produce a single bounding box per face.

B. System Architecture

The system architecture follows a modular design with clear separation of concerns. The main components are:

- **Configuration Module** (`config.py`): Centralizes all hyperparameters and paths
- **Data Parsing** (`csv_parser.py`): Handles annotation loading and dataset statistics
- **Feature Extraction** (`feature_extraction.py`): Implements preprocessing and HOG and LBP algorithms
- **Training Module** (`training.py`): Manages data collection, negative mining, and SVM training
- **Detection Module** (`detector.py`): Performs multi-scale detection with NMS
- **Evaluation Module** (`evaluator.py`): Computes metrics and generates visualizations
- **GUI Module** (`main.py`): Provides user interface for all operations

C. Implementation Flow

The overall system is organized into five main stages: preprocessing, feature extraction, classifier training, detection, and postprocessing. The process begins with input color images that are normalized and filtered to reduce the impact of illumination and noise. From these images, discriminative features are extracted and combined into a unified representation. The SVM classifier is then trained on labeled examples of faces and non-faces. During detection, the trained model scans the image at multiple scales using a sliding window approach to predict face regions. Finally, Non-Maximum Suppression merges overlapping detections and the resulting rectangles are drawn over the detected faces.

1) Training Pipeline:

1) **Data Loading**: Parse CSV annotations mapping images to face bounding boxes
2) **Positive Sample Collection**:
   - Extract face crops from annotated regions
   - Apply horizontal flipping for data augmentation
   - Extract HOG+LBP features (2319 dimensions)
3) **Negative Sample Generation**:
   - Random sampling of 64×64 windows
   - Ensure IoU < 0.1 with all face boxes
   - Filter low-variance regions (std < 3)
   - Collect 25 negatives per face in image
4) **Initial Training**:
   - Standardize features using z-score normalization
   - Train RBF-SVM with C=5.0, gamma='scale'
   - 80/20 train-validation split with stratification
5) **Hard Negative Mining** (2 rounds):
   - Run detector on training images (threshold=0.0)
   - Extract false positive windows as hard negatives
   - Add to training set and retrain
6) **Model Serialization**: Save SVM to pickle file

2) Detection Pipeline:

1) **Image Preprocessing**:
   - Resize if dimension > 640 pixels
   - Preserve aspect ratio
2) **Multi-Scale Detection**:
   - Scale from 1.0 to 0.5 with factor 1.2
   - For each scale:
     - Resize image
     - Slide 64×64 window (stride=12)
     - Extract and normalize features
     - Compute SVM decision function
     - Threshold at 0.5
3) **Post-Processing**:
   - Transform coordinates to original scale
   - Apply NMS with IoU threshold 0.3
   - Return final detections with confidence scores

3) Evaluation Pipeline:

1) **Detection**: Run detector on all test images
2) **Matching**: Greedy assignment of detections to ground truth using IoU ≥ 0.35
3) **Classification**:
   - True Positive: Detection matched with ground truth
   - False Positive: Detection without match
   - False Negative: Ground truth without match

4) **Metrics Computation**:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3}$$

5) **Visualization**: Generate color-coded images showing TP (green), FP (orange), matched GT (blue), missed GT (red)

D. Method Description

1) Feature Extraction: HOG: HOG captures the distribution of local intensity gradients, providing robust shape descriptors. The algorithm consists of:

1) Gradient computation using Sobel operators:

$$G_x = I * S_x, \quad G_y = I * S_y \tag{4}$$

2) Magnitude and orientation calculation:

$$|G| = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan(G_y/G_x) \tag{5}$$

3) Cell-based histogram construction with bilinear interpolation
4) Block normalization using L2-Hys:

$$v' = \frac{v}{\sqrt{||v||^2 + \epsilon^2}} \tag{6}$$

For a 64×64 window with 8×8 pixel cells and 2×2 cell blocks, HOG produces 1764-dimensional feature vectors.

2) Feature Extraction: LBP: LBP encodes local texture through pixel comparisons with circular neighbors:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{7}$$

where $s(x) = 1$ if $x \geq 0$, else 0. Uniform patterns, having at most 2 bitwise transitions, reduce dimensionality while maintaining discriminative power. The resulting histogram provides rotation-invariant texture features, yielding 555 dimensions for P=24, R=3 uniform LBP.

3) Classification: Support Vector Machine: SVM finds the optimal hyperplane maximizing the margin between classes:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \tag{8}$$

subject to $y_i(w \cdot x_i + b) \geq 1 - \xi_i$. The RBF kernel enables non-linear decision boundaries:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{9}$$

The decision function computes:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{10}$$

Classification threshold: $f(x) > \tau$ indicates face detection.

4) Hard Negative Mining: Initial SVM training produces false positives on background regions. Hard negative mining iteratively:

1) Detects faces on training images using current model
2) Identifies false positives (non-face regions classified as faces)
3) Adds these as negative training samples
4) Retrains the classifier with augmented dataset

This bootstrapping approach significantly improves precision by teaching the classifier to reject challenging non-face patterns.

5) Multi-Scale Sliding Window Detection: Detection proceeds through:

1) Image pyramid construction with scale factor 1.2 (scales: 1.0, 0.83, 0.69, 0.58, 0.48)
2) Sliding 64×64 window with stride 12 pixels at each scale
3) Feature extraction (HOG+LBP) for each window
4) SVM classification and decision function thresholding
5) Coordinate transformation back to original image scale

6) Non-Maximum Suppression: NMS eliminates redundant detections through greedy selection:

Algorithm 1 Non-Maximum Suppression

```
Sort detections by confidence score (descending)
while detections remaining do
    Select highest scoring detection $d_{best}$
    Add $d_{best}$ to final output
    Remove all detections with $IoU(d, d_{best}) > \tau_{NMS}$
end while
return final detections
```

Intersection over Union (IoU) measures overlap:

$$IoU(B_1, B_2) = \frac{Area(B_1 \cap B_2)}{Area(B_1 \cup B_2)} \tag{11}$$

Setting $\tau_{NMS} = 0.3$ balances duplicate elimination with preserving nearby faces.

E. Module Descriptions

1) Feature Extraction Module: Implements complete feature extraction pipeline:

- `preprocess_image()`: Grayscale conversion, resizing, CLAHE normalization, Gaussian denoising
- `extract_hog()`: Full HOG implementation with gradient computation, cell histograms, block normalization
- `extract_lbp()`: Circular LBP with bilinear interpolation, uniform pattern detection, histogram generation
- `extract_features()`: Combined HOG+LBP feature vector

Key implementation details:

- Sobel operators for gradient computation
- Bilinear interpolation for sub-pixel neighbor values
- L2-Hys block normalization with clipping at 0.2
- Uniform pattern detection via transition counting

2) Training Module: Manages complete training workflow:

- `collect_training_samples()`: Data augmentation, negative sampling with IoU checking
- `mine_hard_negatives()`: Iterative false positive collection
- `train()`: End-to-end training with hard negative mining rounds
- `save_model()`/`load_model()`: Model persistence

3) Detection Module: Implements efficient multi-scale detection:

- `detect()`: Main detection interface with optional image resizing
- `_detect_internal()`: Multi-scale sliding window search
- `non_max_suppression()`: Greedy NMS based on IoU
- `draw_detections()`: Visualization with bounding boxes

Performance optimizations:

- Downscale large images (max dimension 640)
- Skip small scales when window exceeds image
- Efficient IoU computation
- Confidence-based NMS ordering

4) Evaluation Module: Comprehensive evaluation framework:

- `evaluate()`: Complete test set evaluation
- `calculate_iou()`: IoU computation for matching
- `draw_evaluation_results()`: Color-coded visualization

Outputs:

- Quantitative metrics (precision, recall, F1)
- Per-image visualizations showing all detection outcomes
- Metrics text file for record keeping

F. Algorithm Implementations

1) HOG Feature Extraction: The HOG algorithm processes images through the following steps:

Algorithm 2 HOG Feature Extraction

```
Compute gradients: $G_x, G_y$ via Sobel
Calculate magnitude and angle
Divide image into 8×8 cells
for each cell do
    Create 9-bin gradient histogram
    Use bilinear interpolation for votes
end for
for each 2×2 block of cells do
    Concatenate cell histograms
    Apply L2-Hys normalization
end for
Concatenate all normalized blocks
return feature vector
```

2) LBP Feature Extraction: The LBP algorithm encodes texture patterns as follows:

Algorithm 3 LBP Feature Extraction

```
Compute circular neighbor coordinates
for each pixel do
    $pattern \leftarrow 0$
    for each of P neighbors do
        $value \leftarrow$ bilinear interpolation
        if $value \geq center$ then
            $pattern \leftarrow pattern | (1 << p)$
        end if
    end for
    $LBP(x, y) \leftarrow$ uniform pattern code
end for
Compute histogram of LBP codes
Normalize histogram
return feature vector
```

G. Application Usage

The system provides both GUI and command-line interfaces:

1) Graphical Interface: The main GUI (`main.py`) offers:

- **View Config**: Display current hyperparameters
- **Dataset Info**: Show training/test set statistics
- **Train Model**: Interactive training with progress display
- **Detect Single Image**: Browse and detect on individual images
- **Batch Detection**: Process multiple images from folder
- **Evaluate Model**: Run full evaluation on test set
- **Open Output Folder**: Quick access to results

Features:

- Threaded operations to prevent UI freezing
- Real-time console output with emoji indicators
- Progress bars for long operations
- Side-by-side comparison viewer for detections
- Automatic model loading on startup

2) Command-Line Interface: Individual modules can be run directly:

```
# Training
python training.py

# Evaluation
python evaluator.py

# Single detection
python detector.py
```

3) Configuration: Key parameters in `config.py`:

- Window size: 64×64 pixels
- HOG: 9 orientations, 8×8 cells, 2×2 blocks
- LBP: 24 points, radius 3, uniform
- SVM: RBF kernel, C=5.0
- Detection: threshold=0.5, NMS IoU=0.3
- Evaluation: IoU threshold=0.35

- Training: 400 images max
- Testing: 150 images

IV. Experimental Results

A. Dataset

The system was evaluated on the Human Faces Object Detection dataset from Kaggle [1], which provides annotated face bounding boxes in diverse real-world images.

- **Source**: Kaggle - Human Faces Object Detection Dataset
- **Training Set**: 400 images with face annotations
- **Test Set**: 100 images, 195 ground truth faces
- **Image Properties**: Color images, various resolutions, single/multiple faces
- **Annotations**: CSV format with bounding boxes (x0, y0, x1, y1)
- **Challenges**: Varying lighting, occlusions, pose variations, cluttered backgrounds

The dataset provides realistic challenges for face detection including variations in face size, orientation, lighting conditions, and partial occlusions, making it suitable for evaluating classical computer vision approaches.

B. Detection Performance

Quantitative results on 100 test images:

TABLE I
Detection Performance Metrics

Metric	Value
Ground Truth Faces	195
Detected Faces	163
True Positives	114
False Positives	49
False Negatives	81
Precision	69.94%
Recall	58.46%
F1-Score	63.69%

C. Visual Results

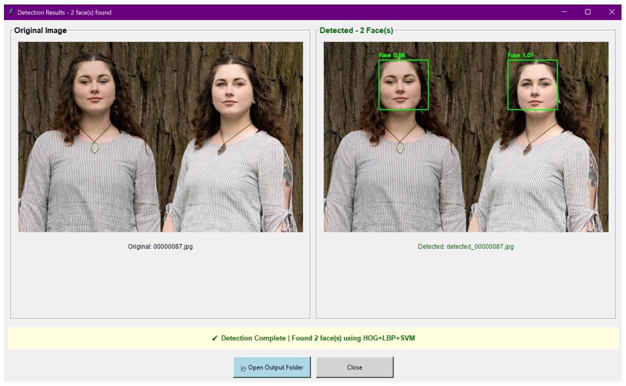
Figure 1 shows a successful detection example on a test image containing two faces. The system correctly identifies both faces with high confidence scores (0.88 and 1.01), demonstrating effective frontal face detection capabilities.

The color-coded evaluation visualizations provide detailed insight into system performance. True positives (green boxes) show correctly detected faces, false positives (orange boxes) indicate spurious detections, matched ground truth (blue boxes) confirm accurate localization, and false negatives (red boxes) highlight missed faces.

D. Qualitative Analysis

1) Strengths: The system demonstrates robust performance in:

- Frontal face detection with high confidence
- Handling illumination variations through CLAHE normalization
- Multi-face detection in group photos
- Scale invariance through pyramid search
- Background clutter rejection after hard negative mining



Original: 00000087.jpg

Detected: detected_00000087.jpg

✓ Detection Complete | Found 2 face(s) using HOG+LBP+SVM

Fig. 1. Detection example: Original image (left) and detected faces with bounding boxes (right). Green boxes indicate successful face detections with confidence scores.

2) Limitations: Observed failure cases include:

- Profile faces
- Severe occlusions
- Very small faces
- Extreme lighting
- Motion blur

V. Conclusion

This project demonstrates that classical computer vision techniques, when carefully implemented and optimized, remain viable solutions for face detection tasks. The combination of well-engineered features (HOG+LBP), robust classification (RBF-SVM with hard negative mining), and multi-scale detection achieves competitive performance while maintaining interpretability and computational efficiency.

A. Achievements

This project successfully implemented a complete face detection system achieving the following objectives:

1) **Robust Feature Extraction**: Implemented from-scratch HOG and LBP algorithms, producing 2319-dimensional feature vectors that capture both shape and texture information
2) **Effective Classification**: Trained RBF-SVM classifier with hard negative mining, achieving 98.51% validation accuracy
3) **Multi-Scale Detection**: Developed efficient sliding window framework with pyramid search, enabling detection across scales from 0.5× to 1.0×
4) **Competitive Performance**: Achieved 69.94% precision, 58.46% recall, and 63.69% F1-score on 100 diverse test images, demonstrating effective face detection using classical methods
5) **Comprehensive Evaluation**: Implemented complete evaluation pipeline with IoU-based matching, detailed metrics, and color-coded visualizations
6) **User-Friendly Interface**: Created intuitive GUI supporting training, detection, batch processing, and evaluation with real-time feedback

All proposed objectives were fully accomplished, with the system demonstrating competitive performance for a classical computer vision approach.

B. Future Development Directions

Several promising directions for future work:

1) **Multi-View Detection**: Extend training data to include profile faces at various angles (0°, ±30°, ±60°, 90°), potentially using multiple specialized classifiers
2) **Pose Estimation**: Integrate head pose estimation to handle rotated faces, possibly through landmark detection
3) **Cascade Architecture**: Implement rejection cascade similar to Viola-Jones to accelerate detection by early rejection of obvious non-faces
4) **Feature Optimization**: Explore dimensionality reduction (PCA, LDA) to reduce feature vector size from 2319 to 500 dimensions for faster classification
5) **Real-Time Processing**: Optimize sliding window search through spatial hashing or GPU acceleration to achieve video frame rate processing
6) **Hybrid Approach**: Combine classical features with lightweight CNN features to balance accuracy and computational efficiency
7) **Soft-NMS**: Replace greedy NMS with Soft-NMS to better handle overlapping faces in crowded scenes
8) **Quality Assessment**: Add face quality scoring to filter low-confidence detections in ambiguous cases
9) **Extended Evaluation**: Test on standard benchmarks (FDDB, WIDER FACE) for direct comparison with state-of-the-art methods

References

[1] S. Baghbidi, "Human Faces Object Detection Dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/sbaghbidi/human-faces-object-detection