Face Mask Detection Using YOLO and CNNs

Kartikey Agarwal

June 25, 2025

1 Introduction

This project focuses on building a deep learning-based model that can detect whether individuals in an image are wearing a mask correctly, incorrectly, or not at all. The model is based on YOLO (You Only Look Once), a popular real-time object detection framework that leverages Convolutional Neural Networks (CNNs) for detecting and classifying objects.

2 Approach

Our approach follows a structured pipeline involving data annotation parsing, image-label conversion to YOLO format, model training, and evaluation. The main stages include:

1. **Data Extraction and Cleaning:** XML annotation files are parsed to extract bounding box and label data.
2. **Normalization:** Bounding box coordinates are normalized relative to image dimensions.
3. **Label Encoding:** Class labels are mapped to integer values.
4. **Formatting:** Data is saved in YOLO-specific format with one text file per image.
5. **Splitting:** Data is split into training and testing sets to evaluate generalization.
6. **Model Selection:** YOLO from Ultralytics is used for training due to its balance of speed and accuracy.
7. **Training:** Images and corresponding label files are fed into the YOLOv8 training pipeline.
8. **Evaluation:** The trained model is assessed using mAP and visual inspection of predictions.

This structured methodology ensures consistent input to the model and effective training and validation.

3 Dataset Description

The dataset includes annotated images sourced from a public dataset available on Kaggle. Each image is associated with an XML file (PASCAL VOC format) that provides bounding box coordinates and class labels:

- `with_mask`
- `mask_weared_incorrect`
- `without_mask`

4 Data Preprocessing

The annotations are parsed using Python's `xml.etree.ElementTree`. For each image, bounding box coordinates are normalized:

- **Center X, Y:** Normalized using image width and height.
- **Width, Height:** Also normalized.

The labels are mapped to numerical values using a dictionary. Images and corresponding label files are then converted into YOLO format for training. A new CSV file is created to store this normalized data, and a mapping from class names to numerical IDs is applied.

5 Train-Test Split

The dataset is randomly split with 80

6 Model Architecture

The model uses YOLO (You Only Look Once), specifically implemented via the `ultralytics` package. YOLO is a fast and accurate real-time object detection system that utilizes CNNs for feature extraction and bounding box prediction. The model accepts image-label pairs as inputs and returns predictions that include object location and category.

6.1 Convolutional Neural Networks (CNNs)

CNNs form the backbone of YOLO and are responsible for learning spatial hierarchies of features from images. A typical CNN consists of the following layers:

- **Convolutional Layers:** Apply filters to extract features like edges, textures, and shapes.
- **Activation Functions:** Typically ReLU, applied to introduce non-linearity.

- **Pooling Layers:** Reduce spatial dimensions while retaining important information.
- **Fully Connected Layers:** Perform high-level reasoning based on extracted features.

In YOLO, the CNN outputs both class probabilities and bounding box coordinates in a single forward pass, making the detection process highly efficient.

7 Training and Evaluation

The model is trained using the YOLO training script provided by the `ultralytics` framework. The command line interface allows specifying hyperparameters such as batch size, number of epochs, and image resolution. During training, images and labels are augmented and fed to the model in batches.

After training, the model is evaluated using metrics such as:

- **Precision and Recall:** Measure accuracy and completeness of detections.
- **mAP (mean Average Precision):** Captures overall detection performance across all classes.
- **Loss Curves:** Visualizations are generated to monitor classification loss, localization loss, and overall training progress.

8 Results and Observations

After training, the model performs well in distinguishing between the three categories. It accurately detects multiple faces per image and classifies them correctly under varied lighting and background conditions. False positives are minimal, and the model generalizes well to unseen data.

Sample predictions show bounding boxes correctly drawn around faces along with corresponding class labels. Training logs and evaluation reports show that the model achieves high mAP and low classification error.

9 Conclusion

This project demonstrates the effective use of YOLO and CNNs for object detection tasks like face mask detection. The model is capable of performing real-time detection with high accuracy, making it suitable for applications in surveillance, access control, and public safety enforcement.

Future improvements could include fine-tuning on more diverse datasets, optimizing the model for mobile devices, and integrating live video feed for real-time applications.