Project Report

Demonstration-video:https://www.loom.com/share/da309a6333944401a8c51071701ac4fa?sid=e842ba3a-aae8-4a23-bd9b-2f555864d4c6

Objective: The goal of this project is to develop a system that can automatically detect individuals and their safety gear in various environments. This system aims to ensure that people are wearing the necessary protective equipment, thereby improving workplace safety and reducing the risk of accidents and injuries.

Problem Statement: Tracking whether everyone is wearing the necessary safety gear can be challenging, especially in busy environments with many people. By leveraging advanced computer vision technology, specifically the YOLOv8 model, this project aims to simplify this process by automatically spotting people and checking their protective equipment in real-time.

Key Components

1. **Data Preparation**
   Annotation Conversion
   To prepare the dataset for training, we need to convert annotations from the PascalVOC format to the YOLOv8 format. PascalVOC annotations are provided in XML files, which contain detailed information about bounding boxes and class labels. YOLOv8 requires annotations in a different format: a plain text file for each image, where each line contains the class ID and bounding box coordinates in relative terms.
   The conversion process involves several steps:
   - **Parsing XML Files:** We use the `xml.etree.ElementTree` library to parse each XML file, extracting relevant information such as bounding box coordinates and image dimensions.
   - **Extracting Image Dimensions:** Each XML file contains a `<size>` element with the width and height of the image. These dimensions are essential for normalizing the bounding box coordinates to ensure they are scaled relative to the image size.
   - **Mapping Class Names to IDs:** We map each class name (e.g., ‘person’, ‘hard-hat’) to a unique integer ID using a predefined dictionary that translates class names into integer identifiers.

- **Calculating YOLO Format:** YOLOv8 requires bounding boxes in the format (class_id, x_center, y_center, width, height). We convert PascalVOC coordinates (xmin, ymin, xmax, ymax) by calculating x_center and y_center as the average of xmin and xmax, and ymin and ymax, respectively. These values are normalized by dividing by the image width and height, and width and height are derived from the differences between xmax and xmin, and ymax and ymin, respectively.
- **Saving YOLO Annotations:** The annotations are saved as `.txt` files, each corresponding to an image and containing one line for each detected object, formatted as class_id x_center y_center width height.

2. This conversion ensures that the annotations are in a format compatible with YOLOv8, facilitating effective training of the detection models.

   Training the Person Detection Model Using YOLOv8

   To build a model that can detect people in images, we followed a structured approach:
   - **Environment Setup:** Installed necessary libraries, including the `ultralytics` package for YOLOv8. This setup allows us to leverage YOLOv8's advanced object detection capabilities.
   - **Dataset Processing:** We created a script called `process_dataset()` to prepare the dataset for training. This script:
     - Copies images and their corresponding labels to a new directory.
     - Filters out annotations to retain only those related to the 'person' class.
   - **Dataset Splitting:** Divided the dataset into training and validation sets using an 80-20 split ratio. The `split_dataset()` function shuffles the images and moves them to respective directories for training and validation.
   - **Configuration File Creation:** Generated a YAML configuration file named `data.yaml` specifying paths to the training and validation datasets, number of classes (1 for 'person'), and class names.
   - **Model Training:** Trained the YOLOv8 model using the processed dataset:
     - Loaded a pre-trained model, `yolov8n.pt`, to speed up convergence and improve accuracy.
     - Set training parameters such as the number of epochs (100) and image size (640).
     - Used the `train()` method from the YOLO class to perform training.

- **Inference and Results:** Tested the model on a sample image from the validation set using the `model()` function. Results were saved in the results directory for further analysis.

3. Filtering annotations was achieved by processing the dataset to only include annotations related to the 'person' class, ensuring that the YOLOv8 model was trained specifically for detecting people.

   Training the PPE Detection Model Using YOLOv8

   Objective: The aim of this phase is to train a YOLOv8 model specifically for detecting various types of personal protective equipment (PPE) on cropped images of individuals. This training enables the system to identify and assess PPE in real-time, contributing to workplace safety.

   Steps Involved:

   - **Setup and Environment Preparation:**
     - Installed required libraries, particularly `ultralytics`, using package managers like pip.
     - Configured the YOLOv8 environment with necessary packages for model training and inference.
   - **Dataset Preparation:**
     - **Data Collection and Annotation:** Gathered a dataset of cropped images where PPE items are labeled. The dataset included different PPE types such as masks, gloves, helmets, etc.
     - **Annotation Conversion:** Converted PascalVOC annotations to YOLOv8 format, creating text files where each line represents an object in the format: class_id x_center y_center width height.
     - **Class ID Adjustment:** Adjusted class IDs by subtracting 1 from all IDs to align with YOLOv8's class indexing.
   - **Dataset Splitting:** Divided the dataset into training and validation subsets using an 80-20 split ratio.
   - **Configuration File Creation:**
     - Created a YAML configuration file (`ppe_data.yaml`) specifying paths to the training and validation datasets, total number of PPE classes (8), and class names.
   - **Model Training:**
     - Loaded pre-trained YOLOv8 weights (e.g., `yolov8n.pt`) for transfer learning.
     - Set training parameters including epochs (100), image size (640x640), and batch size (16).
     - Used the `train()` method from YOLOv8 to execute training.

- **Evaluation and Validation:**
  - Evaluated the model on the validation dataset using metrics such as mean Average Precision (mAP), precision, recall, and F1 score.
- **Inference and Results:**
  - Tested the trained model on new images to verify its performance and saved the model weights for deployment.

4. **Inference for Person and PPE Detection Using YOLOv8**

   Objective: The goal of the inference phase is to utilize the trained YOLOv8 models for detecting people and their protective equipment (PPE) in new images. This phase involves processing images to highlight detections and provide feedback on the presence and type of PPE.

   Steps Involved:
   - **Setup and Environment Preparation:**
     - Ensured necessary libraries were installed, including `opencv-python` for image processing and `ultralytics` for YOLOv8.
   - **Loading Models:**
     - Loaded the trained YOLOv8 models for person detection and PPE detection.
   - **Processing Images:**
     - **Image Loading:** Iterated through images in the input directory, loading each image using OpenCV.
     - **Detection:** Ran the person detection model on each image to locate individuals.
     - **Person Detection:**
       - Extracted bounding boxes for detected persons.
       - Cropped images of detected persons from the original image.
       - Labeled bounding boxes around detected persons on the original image.
     - **PPE Detection:**
       - Used the cropped person images for the PPE detection model.
       - Converted bounding box coordinates from cropped images back to original image coordinates.
       - Drawn bounding boxes around detected PPE items on the original image and added text labels with improved visibility.
   - **Saving Results:**

        - Saved the processed images with overlaid detections in the output directory.
5. **Evaluation Metrics:**
    - **Precision:** Measures the accuracy of the model in detecting true positives among all detected positives.
    - **Recall:** Measures the model's ability to find all relevant instances of objects.
    - **F1 Score:** Combines precision and recall into a single metric, providing a balanced evaluation.
    - **mean Average Precision (mAP):** Aggregates precision scores across different recall levels to assess overall model performance.

Summary: The project involves preparing datasets, training YOLOv8 models for detecting people and PPE, performing inference, and visualizing results. Key tasks included annotation conversion, dataset splitting, model training, and evaluation. The models were evaluated using precision, recall, F1 score, and mAP metrics. The inference phase includes processing images to highlight detections and improve label visibility.

Learnings and

- Hands-On Experience with Annotations:Gained invaluable experience in manipulating annotations for the first time.Learned how to convert annotations into necessary formats for training, solidifying my understanding of the data preparation process.
- Understanding of Evaluation Methods:Developed a deeper understanding of evaluation metrics such as precision, recall, F1 score, and mean average precision (mAP).Familiarized myself with the importance of these metrics for assessing model performance and guiding improvements.
- Importance of Data Quality and Diversity:Recognized that well-annotated and diverse datasets significantly impact model performance.Understood that data quality is a critical factor in applied computer vision.
- Challenges in Object Detection:Explored common challenges such as occlusion, scale variation, and illumination changes in real-world applications of object detection.
- Insights into Advancements in Object Detection Technologies:Learned about the evolution from traditional methods to deep learning techniques in object detection.Gained insights into the significance of frameworks like YOLO and SSD for achieving real-time detection.

- Exposure to Transformer-Based Models:Discovered how transformer-based models are setting new standards in performance within the field of object detection.
- Exploration of Libraries and Frameworks:Inspired to explore various libraries and frameworks, enhancing my technical skill set.Prepared for a career in applied computer vision through practical experience and theoretical knowledge.

Future Idea:

- Implement Anomaly Detection: I could explore integrating anomaly detection algorithms to identify unusual patterns in PPE usage, such as workers entering restricted areas without proper gear. This could enhance safety protocols and compliance monitoring, allowing for proactive measures to be taken in real-time.
- Integrate with IoT Devices: I could explore integrating the object detection system with Internet of Things (IoT) devices to create a smart safety monitoring system. For example, wearable devices could alert workers when they are not wearing the required PPE, and the system could log compliance data for further analysis.
- Develop a Safety Compliance Dashboard: I could create a user-friendly dashboard that visualizes compliance data, trends, and alerts in real-time. This dashboard could provide insights into PPE usage patterns and help safety managers make informed decisions, showcasing the practical applications of computer vision in industry.
- Explore Federated Learning: I could investigate federated learning techniques to train the model across multiple devices without sharing sensitive data. This approach would enhance privacy while still benefiting from diverse datasets, making it suitable for applications in industries with strict data regulations.
- Conduct Field Trials: Implementing field trials in actual workplace environments would provide valuable feedback and insights for further improvements. Gathering real-world data would help refine the model and ensure its effectiveness in practical applications, bridging the gap between theory and practice.