Helmet Detection

Tags	DONE
Date	@April 20, 2023
Links	https://github.com/Maaitrayo/Safety-Helmet-Detection-yoloV8
# Number	3

Project Objective:

The project involved creating a script for detecting helmets in images and videos using the YOLOv8 object detection algorithm. The script was designed to take a folder path as input and detect helmets in all the images and videos contained within that folder. The script then saved the annotated images and a CSV file containing information about the detections in a separate output folder.

Tools Used:

1. Python Programming Language
2. OpenCV (Open Source Computer Vision Library) - to work with images and videos
3. YOLOv8 (You Only Look Once) Model - to detect objects in images and videos
4. Supervision (Python Package) - to visualize object detection and annotations
5. Ultralytics (Python Package) - to use the YOLO model

Project Workflow:

1. Load the YOLO model that has been trained to detect helmets.
2. Read the input video or images and resize the frames to the required size.
3. Pass the resized frames through the YOLO model to get the detected objects and their positions.
4. Use the Supervision package to visualize the detections on the image.
5. Store the resulting images with the annotations in a folder.

6. Extract the labels of the detections from the YOLO results.
7. Check whether each person in the image is wearing a helmet or not, and store the results in a CSV file.
8. Store the CSV file in a folder along with the images.

Functions Used:

1. show_file_size():
    - The `show_file_size()` function takes a file path as input and prints its size in megabytes. It uses the `os.path.getsize()` method to get the size of the file in bytes and converts it to megabytes. The function rounds the file size to two decimal places and then prints it to the console. This function is used to get an idea of the size of the images and the CSV file generated by the program.
2. imageLoader():
    - The `imageLoader()` function takes a folder path as input and returns a tuple containing two lists: a list of all image paths within the folder and a list of their corresponding names. It uses the `os.listdir()` method to get a list of all items in the folder and then loops through each item to get the path of the item. The function then returns a tuple containing the list of image paths and the list of image names. This function is used to get the paths of all the images in a folder.
3. saveResultCSV():
    - The `saveResultCSV()` function takes a list of results, an output folder name, and a CSV file name as input and saves the results to a CSV file in the specified output folder. It combines the output folder name and CSV file name to form the full path to the CSV file using the `os.path.join()` method. It then uses the `csv.writer()` method to create a CSV writer and writes the header row to the CSV file. The function then loops over the results and writes each row to the CSV file. This function is used to save the results of the program to a CSV file.
4. checkHeads():
    - The `checkHeads()` function checks if the given list of labels contains the word "head". If it does, the function saves the corresponding image to disk and adds a row to the result list indicating that no helmet was detected. The

function takes a list of labels detected in the image, a list of all image names in the folder, a list of all image paths in the folder, the image array, the current list of results, the index of the current image, and the path of the folder where the images will be saved as input. If the label "head" is in the list of labels, the function gets the image name and location and saves the image to disk using the `cv2.imwrite()` method. The function then generates the message to store in the CSV file and adds it to the list of results. This function is used to check if a person in an image is wearing a helmet or not and to update the results accordingly.

5. processImages():
    - This function takes a list of image paths, a list of corresponding image names, and an output folder path as inputs. It resizes the images, detects helmets in the images using a pre-trained YOLOv8 model, annotates the images with bounding boxes around the detected helmets, checks if the detected helmets are properly worn, and adds the detection result to a list. Finally, it saves the annotated images and detection results to the output folder and returns the detection results as a list.
6. m**ain()**:
    - This is the main function of the program. It parses command line arguments to determine the input folder containing images to be processed, and the output folder where annotated images and detection results are to be stored. It calls the `processImages` function to perform helmet detection on the input images, and saves the detection results to a CSV file.

Limitations and Potential Improvements:

1. The model may not be accurate in all situations, and there may be false positives and false negatives. One way to improve the accuracy is to fine-tune the model on a larger and more diverse dataset.
2. The current implementation only detects helmets, but it could be extended to detect other safety equipment such as safety glasses or gloves.
3. The current implementation only works with images and videos, but it could be extended to work with live camera feeds.

Conclusion:

In conclusion, your project involved detecting helmets in images and videos using a YOLO model. You used Python, OpenCV, YOLO, Supervision, and Ultralytics to implement the solution. The project workflow involved loading the YOLO model, reading the input images or video frames, passing them through the model, visualizing the detections, checking whether each person is wearing a helmet or not, and storing the results in a CSV file. There are potential improvements that could be made to the project, but overall it provides a good foundation for detecting safety equipment in images and videos.