SYOOK AI INTERNSHIP ASSIGNMENT

Problem Statement: Syook_AI_Internship_Assignment

Folders and Files

1. **.venv** : virtual environment with python 3.12.2.

2. **datasets** : contains the datasets

    - `voc_original` : dataset given in problem statement
    - `yolo_annotations` : annotations in yolo format generated from voc_original labels (pascalVOC_to_YOLO.py)
    - `yolo_orignal` : dataset in yolo format from voc_original images and yolo_annotations (create_original_dataset.py)
    - `yolo_persons` : dataset in yolo format with only person class from yolo_original dataset (create_person_dataset.py)
    - `yolo_ppe` : dataset in yolo format with only ppe classes from yolo_original dataset (create_ppe_dataset.py)
    - `Person_Detetction.v1i.yolov8` : dataset for person detection model from yolo_person dataset uploaded to roboflow here.
    - `PPE_Detection.v1i.yolov8` : dataset for ppe detection model from yolo_ppe dataset uploaded to roboflow here.
      This dataset only has six classes a the other ppe item classes are not present in the original dataset. They are:

```
ppe_classes = [
    'boots',
    'gloves',
    'hard-hat',
    'mask',
    'ppe-suit',
    'vest'
]
```

3. **test** : contains test folders for all generated datasets and also for inference

    - `inference` : contains input images for inference and output images after inference (inference.py)
    - `yolo_original` : test folder for yolo_original dataset
    - `yolo_person` : test folder for yolo_person dataset
    - `yolo_ppe` : test folder for yolo_ppe dataset

4. **utils** : contains utility functions

    - `tests.py` : contains function to be used to test the generated datasets in test folders

- `utils.py`: contains function to split a geiven dataset into train, test and validation sets
- `vars.py` : contains global variables

5. **weights** : contains weights for all the generated models
    - **Person** : weights for Person detection models
        - `person_50_original` : best.pt for person detection model trained on yolo_person v1 dataset 50 epochs.
        - `person_100_resized` : best.pt for person detection model trained on yolo_person v2 dataset 100 epochs.
        - `person_200_original` : best.pt for person detection model trained on yolo_person v1 dataset 200 epochs.
    - **PPE** : weights for PPE detection models
        - `ppe_50_original` : best.pt for ppe detection model trained on yolo_ppe v1 dataset 50 epochs.
        - `ppe_100_resized` : best.pt for ppe detection model trained on yolo_ppe v2 dataset 100 epochs.
        - `ppe_200_original` : best.pt for ppe detection model trained on yolo_ppe v1 dataset 200 epochs.

Process and Approach

Note: For all the args the default values are already set so that the scripts can be run without any args unless chnages are required.

1. Converting Pascal VOC format to YOLO format -> pascalVOC_to_YOLO.py
    - takes folder with Labels and classes.txt file, and an output folder and converts the labels to yolo format indexed with classes.txt and saves them in the output folder.
2. Wrote utility functions to split the dataset into train, test and validation sets -> utils.py
    - takes a dataset folder and splits it into train, test and validation sets with `sklearn train_test_split`.
    - also wrote test utilities to run tests on the generated datasets.
3. Creating YOLO dataset from voc_original dataset and converted YOLO annotations -> create_original_dataset.py
    - takes folder with images and yolo annotations and creates a dataset with images and labels in yolo format and save them with `split_dataset` function.
    - test annotating with labels on 10 images at test/yolo_original.
4. Creating YOLO dataset with only person class from yolo_original dataset -> create_person_dataset.py
    - takes yolo_original dataset and creates a dataset with only person class and saves them with `split_dataset` function.
    - test annotating with labels on 10 images at test/yolo_person.

5. Creating YOLO dataset with only ppe classes from yolo_original dataset -> create_ppe_dataset.py

- Approaches tried:
  - trim the cropped images hedges on all sides by x% and then label to prevent annotations from nearby persoon or objects in crop, failed as it is missing some annotations from the original person in the cropped image.
  - several other variations and modifications to the above approach, but failed to get a good dataset.
  - finally, realized that a person only has a limited number of ppe items logically per class so hardcoded them to and then used them to find the closest labels to the center and used only those labels for the cropped image.

```
ppe_max_count = {
    'person': 1,
    'hard-hat': 1,
    'gloves': 2,
    'mask': 1,
    'glasses': 1,
    'boots': 2,
    'vest': 1,
    'ppe-suit': 1,
    'ear-protector': 2,
    'safety-harness': 1
}
```

  - test annotating with labels on 10 images at test/yolo_ppe.

6. Training YOLO models (linked already)

- The datasets of person and ppe were first uploaded to roboflow, so that thye can be easily used in a colab GPU notebook to train with ultralytics yolov8.
- The dsatasets also have a v2 where the images are resized to 640x640 to see if it improves the model performance. The v1 datasets are the original images.
- In case of Person detection the resizing did not show any improvement in the model performance. But in PPE detction since the input images are small in size the resizing performed worse than the original images.
- For final training the v1 datasets were used and trained for 200 epochs for both person and ppe detection models.

7. Infernece on input images -> inference.py

- takes a model and a input folder and runs inference on the images in the test folder and saves the images in the output folder.

8. Evaluate the models with ultraylytics `val` function -> evaluate.py

- The models were evaluated with the `val` function in ultralytics yolov8 to get the mAP and other metrics saved at runs/detect.

Conclusion and Improvements to be made

1. The dataset is very `niche` in images and it does not have annotations for all the ppe items (only `6 classes` are present without person) and more quality and quantity can imporve the model performance significantly. Just addiing more images with variations and deleting some of the false positives could have made a huge impact and done on `roboflow` itself, but due to the time constraint it was not done.

2. The Person detetction is glitched sometimes and detects the same person more than once, and this only went up with `epochs` increase, but it also picked up persons in the background which were missed initially.

3. The PPE detection works well with the cropped people and is able to diffferentaite multiple people with different ppe items, but it also picks up some false positives which can be solved by setting a higher `confidence threshold`.

4. Overall the experience was good and i focused more on making the code `modular` and reusable and also tried to make the code as clean as possible. The code is also well `documented` to make it easy to understand what is happening at each step.