YOLOv5 Model Training and Evaluation

Dataset Configuration for YOLO Training

Before training the YOLO model, a configuration file (data.yaml) was created to define the dataset structure. This file was designed to include:

- Train Dataset Path: data/images/train.
- Validation Dataset Path: data/images/test.
- Number of Classes (nc): 20.
- Class Names: Spartan, car, chair, bottle, etc.

By specifying these details, the dataset location and class labels were properly set up for the training process.

Setting Up YOLOv5 Repository

To begin training, the Ultralytics YOLOv5 repository was used, as it provided pre-configured functionalities. The setup process involved the following steps:

1. Mounting Google Drive:
   - Google Drive was mounted by selecting "Files" and choosing "Mount Drive".
   - Access to the drive was granted, allowing the dataset and necessary files to be stored.
2. Navigating to the Project Directory:
   - The os module was imported, and the working directory was changed to the YOLO training folder.
   - The correct path was copied and used for navigation.
3. Cloning the YOLOv5 Repository:
   - The repository was cloned using Git.
   - A new folder named yolov5 was created in Google Drive, containing all necessary scripts and configurations.
4. Uploading Dataset and YAML File:
   - The dataset folder, containing training and validation images, was uploaded.
   - The data.yaml file, which was created earlier, was also uploaded.
   - The upload process took approximately 1.5hrs to 2hrs.

With these steps completed, the YOLOv5 repository and dataset were successfully set up for training.

Training the YOLOv5 Model

- The Google Colab notebook was connected to a GPU runtime by selecting the hardware accelerator option and saving the changes.
- Google Drive was mounted after GPU initialization.
- The directory was changed to YOLOv5 using the cd command, and its contents were verified using the ls command.
- All necessary packages were installed by running the command !pip install -r requirements.txt, ensuring that the correct version of PyTorch and other dependencies were installed.
- The YOLOv5 training process was started using the train.py script. Pre-trained weights were used, and the image size was set to 640. The batch size was reduced to 8, and the model was trained for 40 epochs.
- The system was kept on throughout the training process, and precautions were taken to prevent sleep mode.
- After training, the model was saved in the runs/train folder in Google Drive, with the best weights stored under the weights folder.

- The model was exported as an .onnx file using the export.py script, and it was stored in the runs/train/weights folder.
- The final trained model was downloaded as a .zip file for further use.

YOLOv5 Model Evaluation Report

Model Summary

- Total Layers: 157
- Total Parameters: 7,064,065
- Gradients: 0
- Computational Complexity: 15.9 GFLOPs

Performance Metrics

- Dataset: 1009 images
- Total Instances: 3214
- Precision (P): 0.734
- Recall (R): 0.662
- Mean Average Precision (mAP@50): 72.4%
- Mean Average Precision (mAP@50-95): 45.9%

Class-wise Performance

Class	Instances	Precision (P)	Recall (R)	mAP@50	mAP@50-95
Person	1153	0.765	0.798	0.834	0.497
Car	340	0.816	0.821	0.864	0.592
Chair	264	0.649	0.561	0.569	0.338
Bottle	104	0.554	0.567	0.571	0.363
Potted Plant	144	0.624	0.530	0.530	0.263
Bird	122	0.679	0.566	0.654	0.367
Dog	120	0.730	0.586	0.690	0.461
Sofa	75	0.545	0.431	0.511	0.342
Bicycle	86	0.802	0.791	0.841	0.534
Bus	71	0.838	0.732	0.822	0.616

- The highest performance was observed in Car (86.4%), Train (80.5%), and Sheep (84.6%).
- Lower performance was observed in Sofa (51.1%), Potted Plant (53.0%), and Bottle (57.1%), suggesting areas for improvement.

Next Steps

- Recall improvement for underperforming categories through dataset balancing or augmentation techniques.
- Hyperparameter optimization to enhance detection accuracy.
- Fine-tuning on low-performing classes by increasing training samples or adjusting anchor sizes.
- Testing different backbones to assess their impact on precision and recall.

The results were saved in the runs/train/Model2_continued directory.

YOLO Prediction Process

1. Library Installation and Import
   - Required libraries such as OpenCV, NumPy, OS, and YAML were installed and imported.
   - Since YAML was not installed, it was installed using pip install pyyaml.
2. Loading YAML and YOLO Model
   - The YAML file was loaded to retrieve configuration settings.
   - The YOLO model was loaded for object detection.
3. Image Processing and Predictions
   - An image was loaded and passed through the YOLO model.
   - Detections were extracted from the model's output.
4. Non-Maximum Suppression (NMS)
   - Multiple bounding boxes were filtered using Non-Maximum Suppression (NMS) to keep only the most accurate detections.
5. Drawing Bounding Boxes
   - Final bounding boxes were drawn on the image.

Conclusion

The process of YOLOv5 model training, evaluation, and prediction was successfully completed. The trained model demonstrated high performance on most object categories. Further improvements can be made by fine-tuning the model on underperforming classes, optimizing training parameters, and experimenting with different backbones to enhance detection accuracy.

Reference:

An AI assistant was utilized for paraphrasing, grammar, and sentence structure refinement.