YOLOv5 Custom Training Guide

This guide provides step-by-step instructions for training a YOLOv5 object detection model on a custom dataset using Google Colab or a local environment.

1. Clone the YOLOv5 Repository
First, clone the YOLOv5 repository from Ultralytics:

```bash
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!git reset --hard 064365d8683fd002e9ad789c1e91fa3d021b44f0  # Use a stable commit version
```

2. Install Dependencies
Ensure all required dependencies are installed:

```bash
!pip install -qr requirements.txt  # Install required packages
```

3. Download and Prepare the Dataset
Use Roboflow to fetch and prepare the dataset:

```bash
!pip install roboflow
```
```python
from roboflow import Roboflow
rf = Roboflow(api_key="YOUR_API_KEY")
project = rf.workspace("yollolabel").project("hard-hat-sample-hacx2")
version = project.version(2)
dataset = version.download("yolov5")
```

4. Train the YOLOv5 Model
Train the model for 100 epochs:

```bash
%cd /content/yolov5/
!python train.py --img 640 --batch 20 --epochs 100 --data {dataset.location}/data.yaml --cfg ./models/yolov5s.yaml --weights '' --name yolov5s_results --cache
```

5. Evaluate Model Performance
Plot and display training results:

```python
from utils.plots import plot_results  # Plot results
Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=1000)
```

6. Run Inference on Test Images
Run object detection using the trained model:

```bash
!python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 640 --conf 0.4 --source /content/yolov5/Hard-Hat-Sample-2/test/images
```

7. Improving Model Accuracy
To improve model accuracy, consider the following tips:
- **Increase Dataset Size**: Collect and annotate more images to enhance model learning.
- **Augment Data**: Use transformations like flipping, rotation, and color adjustments to increase dataset diversity.
- **Fine-Tune Hyperparameters**: Adjust learning rate, batch size, and epochs to find optimal values.
- **Use a Pretrained Model**: Start training from a well-trained YOLOv5 model rather than from scratch.

- **Filter Low-Quality Data**: Remove incorrectly labeled or blurry images from the dataset.

8. Handling Insufficient RAM Issues
If training stops due to memory constraints, try the following solutions:
- **Reduce Batch Size**: Lowering the batch size can significantly reduce RAM usage.
- **Use Mixed Precision Training**: Enable `--half` flag in `train.py` to use FP16 precision.
- **Increase Swap Space**: Add swap memory in Google Colab or local machines.
- **Use a Smaller Model**: Consider training with `yolov5s.yaml` instead of larger variants.
- **Train on Cloud Services**: Use platforms like Google Colab Pro, AWS, or Google Cloud with high-end GPUs.

Conclusion
This guide provides a straightforward process to train and test a YOLOv5 model on a custom dataset. You can modify hyperparameters and dataset configurations to improve model performance. Follow the additional tips for enhancing accuracy and overcoming hardware limitations.