Soham Naik – 01/04/2024 – Adaptive HCI [Xiao Zhang, Alan Raj, Deniz Acikbas, Zaynab Mourtada]

YOLO Workflow for Custom Object Detection

This guide provides a step-by-step timeline for training a custom YOLO model to detect an object of interest:

1. Labeling images locally using LabelImg.
2. Setting up the environment on the HPC (Great Lakes).
3. Transferring data between your local machine and the HPC.
4. Training the YOLO model on the HPC using SLURM.
5. Using the trained model in Android Studio.

Contents

Soham Naik – 01/04/2024 – Adaptive HCI [Xiao Zhang, Alan Raj, Deniz Acikbas, Zaynab Mourtada]

Phase 0: Extracting & Preparing The Data

Step 0.1 Modify the Arduino Code

1. Open the Arduino code titled **“optimized_glove”** from GitHub
2. Scroll to the **generateOOKSignal()** function at the bottom of the code.
3. Refer to the provided table and insert your unique Arduino code into the **emitSymbol()** call.
    - Alternatively, comment out the existing line and uncomment your custom line.
4. Upload the modified code to the Arduino Micro.

YOLO/LabelImg Label	Arduino Code	Responsible
User_1	10001000	Soham Naik
User_2	11001100	Deniz Acikbas
User_3	10101010	Zaynab Mourtada
User_4	11110000	Alan Raj

```
void generateOOKSignal() {
  // Emit pilot Gap
  emitPilotGap();
  // Emit symbols as per OOK encoding:
  emitSymbol("10001000");  // User_1 // Soham
  //emitSymbol("11001100");  // User_2 // Deniz
  //emitSymbol("10101010");  // User_3 // Zaynab
  //emitSymbol("11110000");  // User_4 // Alan
}
```

Step 0.2 Configure Xamera on Your Mobile Device

1. Open the **Xamera app** on your mobile device.
2. Adjust the shutter rate to **1,000 Hz (1K Hz).**

Soham Naik – 01/04/2024 – Adaptive HCI [Xiao Zhang, Alan Raj, Deniz Acikbas, Zaynab Mourtada]

Step 0.3 Record Video

1. Start recording a **10-minute video** using Xamera.
    - The recording will automatically begin when you select **"Start Tracking"** and will save upon selecting **"Stop Tracking."**
2. Ensure the video captures diverse scenarios to maximize data quality. Consider variations in:
    - Distance between the phone and the LED.
    - Rotation and angle of the phone relative to the LED.
    - Ambient lighting conditions (e.g., bright windows, indoor lighting, or darkness).
3. Plan your recording carefully to include a wide range of unique situations.

Step 0.4 Transfer the Video

1. Move the saved video file from your mobile device to your laptop.

Step 0.5 Extract Frames from the Video

1. Use a tool like **FFmpeg** or a Python script with **OpenCV** to split the video into individual frames.

```
import cv2
import os
def extract_frames(video_path, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    print(f"Total frames in the video: {frame_count}")
    frame_idx = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Finished extracting frames.")
            break
        frame_filename = os.path.join(output_folder, f"frame_{frame_idx:04d}.jpg")
        cv2.imwrite(frame_filename, frame)
        frame_idx += 1
        if frame_idx % 100 == 0:  # Log every 100 frames
            print(f"Extracted {frame_idx}/{frame_count} frames...")
    cap.release()
    print(f"All frames are saved in {output_folder}")
video_path = "path/to/your/video.mp4"
output_folder = "extracted_frames"
extract_frames(video_path, output_folder)
```

Step 0.6 Select High-Quality Images

1. Review the extracted frames and manually select **100 high-quality images** that represent diverse scenarios.

Soham Naik – 01/04/2024 – Adaptive HCI [Xiao Zhang, Alan Raj, Deniz Acikbas, Zaynab Mourtada]

Phase 1: Labeling Images Locally

Step 1.1: Install LabelImg

LabelImg is a graphical image annotation tool that supports YOLO format. Install it on your local machine:

- GitHub Repository: https://github.com/HumanSignal/labelImg.

On Linux:

- pip install labelImg
- labelImg # Launch the application

On Windows:

1. Download the precompiled executable from https://github.com/HumanSignal/labelImg/releases.
2. Run the executable to launch the application.

Step 1.2: Label the Images

1. Organize your images:
    - Place all images in a folder (e.g., images/).
2. Open LabelImg:
    - Set the image directory to your images/ folder.
    - Set the label directory to a new folder (e.g., labels/) where the .txt files will be saved.
3. Set LabelImg to YOLO mode:
    - In LabelImg, go to View > Auto Save Mode, and ensure the format is set to YOLO.
4. Set predefined classes:
    - In data/predefined_classes.txt, define your class (e.g., your_object_name).
5. Label the images:
    - Use the Create RectBox tool to draw bounding boxes around your object of interest.
    - Assign the class name ***(refer to table)***.
    - Save the annotations. Each image will have a corresponding .txt file in YOLO format:
        - <class_id> <x_center> <y_center> <width> <height>
6. Verify the labels:
    - Ensure each image has a corresponding .txt file with accurate bounding box information.

Step 1.3: Organize the Dataset

Organize the labeled dataset into the following structure:

```
dataset/
|── images/
|   |── train/     # Training images
|   └── val/       # Validation images
└── labels/
    |── train/     # Labels for training images
    └── val/       # Labels for validation images
```

- Split the dataset (80% for training, 20% for validation).
  - You can select these however you'd like; it doesn't matter which images/label data are used for training or validation, just split them!
- Ensure each image in images/train has a corresponding .txt file in labels/train.

Phase 2: Setting Up the HPC Environment

Step 2.1: Access the HPC

- Connect to the VPN (if working remotely):
  - Use https://its.umich.edu/enterprise/wifi-networks/vpn/getting-started.
- Access Great Lakes:
  - Via local terminal: ssh your_username@greatlakes.arc-ts.umich.edu
  - Or via the web: https://greatlakes.arc-ts.umich.edu/.

Step 2.2: Set Up the Virtual Environment

- Create a virtual environment:
  - python -m venv ~/YOLO-Soham
- Activate the virtual environment:
  - source ~/YOLO-Soham/bin/activate
- Load required modules:
  - module load python/3.12.1
  - module load cuda/12.1.1

Step 2.3: Install Dependencies

- Install Ultralytics YOLOv8:
    - pip install ultralytics
- Install PyTorch:
    - pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121

Phase 3: Transferring Data to the HPC

Step 3.1: Transfer Files

- Use scp to transfer the dataset (from a local terminal):
    - scp C:\Users\ your_username@greatlakes.arc-ts.umich.edu:/home/user/
    - scp -r: (flag to transfer folders)

Step 3.2: Verify the Dataset

Check the dataset structure on the HPC:

```
dataset/
|── images/
|   |── train/
|   └── val/
└── labels/
    |── train/
    └── val/
```

Soham Naik – 01/04/2024 – Adaptive HCI [Xiao Zhang, Alan Raj, Deniz Acikbas, Zaynab Mourtada]

Phase 4: Training the YOLO Model on the HPC

Step 4.1: Create the Configuration File

- Create custom_dataset.yaml:
  - nano custom_dataset.yaml
- Add the following content:

```
train: /path/to/dataset/images/train
val: /path/to/dataset/images/val
nc: 1  # Number of classes
names: ['your_object_name']  # Class names
```

Step 4.2: Create a SLURM Job Script

- Create yolo_job.slurm:
  - nano yolo_job.slurm
- Add the following content:

```
#!/bin/bash
#SBATCH --job-name=yolo_cuda_test
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH --time=00:10:00
#SBATCH --mem=16G
#SBATCH --output=yolo_cuda_test.out

module load python/3.12.1
module load cuda/12.1.1
source ~/YOLOv11-Soham/bin/activate

yolo detect train data=custom_dataset.yaml model=yolov8x.pt epochs=100 imgsz=640
```

Model	Size (Parameters)	Speed (ms)	Accuracy (mAP)	Use Case
YOLOv8n	~3.2M	~6.3ms	~37.3	Edge devices, real-time apps
YOLOv8s	~11.2M	~6.4ms	~44.9	Lightweight applications
YOLOv8m	~25.9M	~8.2ms	~50.2	General-purpose detection
YOLOv8l	~43.7M	~10.1ms	~52.9	High-accuracy tasks
YOLOv8x	~68.2M	~12.3ms	~53.9	Maximum accuracy, high-resources

---

Step 4.3: Submit the SLURM Job

- Submit the job:
  - sbatch yolo_job.slurm

Step 4.4: Monitor the Job

- Check job status:
  - squeue -u your_username
- View logs:
  - cat yolo_cuda_test.out

Phase 5: Understanding the Output

- Training Logs:
  - Epoch progress, loss values, and metrics (precision, recall, mAP).
- Model Output:
  - The trained model (best.pt) is saved in:
    - runs/detect/train/

Phase 6: Transferring Results Back to Local Machine

Step 6.1: Transfer Trained Model

- Transfer the trained model to your local machine (from a local terminal):
  - scp your_username@greatlakes.arc-ts.umich.edu:/home/user/runs/detect/train/best.pt C:\Users\
  - scp -r: (flag to transfer folders)

Phase 7: Exit

To exit the HPC:

exit
