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

[Phase 1: Labeling Images Locally 2](#_Toc186847762)

[Step 1.1: Install LabelImg 2](#_Toc186847763)

[Step 1.2: Label the Images 2](#_Toc186847764)

[Step 1.3: Organize the Dataset 3](#_Toc186847765)

[Phase 2: Setting Up the HPC Environment 3](#_Toc186847766)

[Step 2.1: Access the HPC 3](#_Toc186847767)

[Step 2.2: Set Up the Virtual Environment 3](#_Toc186847768)

[Step 2.3: Install Dependencies 4](#_Toc186847769)

[Phase 3: Transferring Data to the HPC 4](#_Toc186847770)

[Step 3.1: Transfer Files 4](#_Toc186847771)

[Step 3.2: Verify the Dataset 4](#_Toc186847772)

[Phase 4: Training the YOLO Model on the HPC 5](#_Toc186847773)

[Step 4.1: Create the Configuration File 5](#_Toc186847774)

[Step 4.2: Create a SLURM Job Script 5](#_Toc186847775)

[Step 4.3: Submit the SLURM Job 6](#_Toc186847776)

[Step 4.4: Monitor the Job 6](#_Toc186847777)

[Phase 5: Understanding the Output 6](#_Toc186847778)

[Phase 6: Transferring Results Back to Local Machine 6](#_Toc186847779)

[Step 6.1: Transfer Trained Model 6](#_Toc186847780)

[Phase 7: Exit 6](#_Toc186847781)

# 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 (e.g., 0).
* Save the annotations. Each image will have a corresponding .txt file in YOLO format:
  + <class\_id> <x\_center> <y\_center> <width> <height>

1. 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 (e.g., 80% for training, 20% for validation).
* 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](mailto: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/](mailto: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/

# 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](mailto: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