Training Custom Dataset Using YOLOX

# 1. Overview

This guide provides a comprehensive, step-by-step explanation for training a custom dataset using YOLOX—an advanced object detection model. It is tailored for practical use cases such as abandoned object detection and assumes the use of YOLOX-S for simplicity. The guide walks you through environment setup, data preparation, training, evaluation, and deployment.

# 2. Environment Setup

Start by creating a dedicated conda environment with Python 3.10 and installing all required dependencies with GPU support (MY CUDA - 12.6).

## 2.1 Create Conda Environment

bash

conda create --name yolox python=3.10 -y

conda activate yolox

## 2.2 Clone YOLOX GitHub Repository

bash

git clone <https://github.com/Megvii-BaseDetection/YOLOX.git>

cd YOLOX

## 2.3 Install Dependencies for GPU

Bash

pip install -v -e .

( If CUDA related any issue occurs then try uninstalling “torch”, “torchvision” and “torchaudio”. Then try downloading it using the below command for CUDA 12.6. or Refer to official Pytorch website )

pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu126

# 3. Dataset Preparation

( The following Docs are made based on COCO format dataset. And not on PASCAL VOC, you need to do some research about it yourself. I have used LVIS DATSET for training the model which provides a format “SIMILAR” to COCO format and not exactly COCO format, so necessary changes had to be made. If you encounter any error regarding “file\_name” then check your json file if it contains any key named with “file\_name”. If not then take the basename of COCO URL and replace it in yolox/tools/demo.py file )

YOLOX requires COCO format. If your dataset is in Pascal VOC, convert it using the following method:

## 3.1 Convert VOC to COCO

bash

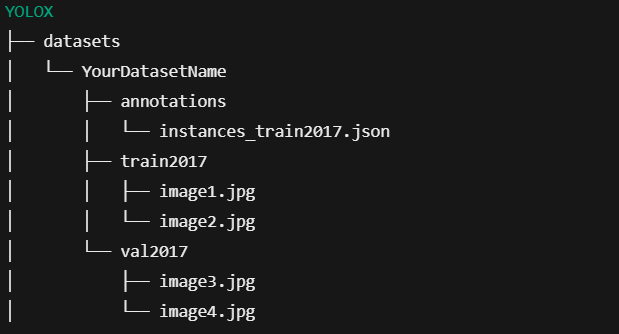
git clone https://github.com/yukkyo/voc2coco.git

cd voc2coco

python voc2coco.py --ann\_dir <path\_to\_annotations> --ann\_ids <path\_to\_ids> --labels <path\_to\_labels> --output <output\_json>

## 3.2 Dataset Folder Structure

( please maintain this dataset structure with the directory and file names too, within YOLOX directory so avoid any issues in future )



# 4. YOLOX Experiment Configuration

Create a new experiment configuration file based on `yolox\_m.py` (here I have used yolox\_m, you could use any of your choice) and customize paths, number of classes, and hyperparameters.

# 5. Training the Model

Train your custom dataset using the command:

bash

python tools/train.py -f exps/default/exp\_my\_dataset.py -d 1 -b 8 --fp16 -c yolox\_s.pth

# 6. Evaluating the Model

Evaluate using the best checkpoint:

bash

python tools/eval.py -f exps/default/exp\_my\_dataset.py -c YOLOX\_outputs/exp\_my\_dataset/best\_ckpt.pth -b 8 -d 1 --conf 0.001

# 7. Running Inference

Run inference on images or webcam being in the YOLOX directory:

\*\*Single Image:\*\*

bash

python tools/demo.py image -f exps/default/base\_exp.py -c YOLOX\_outputs/base\_exp/best\_ckpt.pth --path image.jpeg --conf 0.3 --nms 0.65 --tsize 640 --save\_result --device gpu

( change "image.jpeg" to your image file and and edit the path if necessary also correct the ".jpeg" to ".jpg" or any other as per needs )

\*\*Single Video:\*\*

bash

python tools/demo.py video -f exps/default/base\_exp.py -c YOLOX\_outputs/base\_exp/best\_ckpt.pth --path video.mp4 --conf 0.1 --nms 0.65 --tsize 640 --save\_result --device gpu

( change "video.mp4" to your video file and and edit the path if necessary )

\*\*Webcam:\*\*

bash

python tools/demo.py webcam -f exps/default/exp\_my\_dataset.py -c YOLOX\_outputs/exp\_my\_dataset/best\_ckpt.pth --conf 0.3 --nms 0.65 --tsize 640 --device gpu

# 8. Visualizing Metrics with TensorBoard

Launch tensorboard to inspect training metrics:

bash

tensorboard --logdir YOLOX\_outputs/exp\_my\_dataset

Visit `http://localhost:6006`.

# 9. Export Model for Deployment

Convert your model for deployment:

\*\*ONNX:\*\*

bash

python tools/export\_onnx.py -f exps/default/exp\_my\_dataset.py -c YOLOX\_outputs/exp\_my\_dataset/best\_ckpt.pth --output-name yolox.onnx

\*\*TensorRT:\*\*

bash

python tools/trt.py -f exps/default/exp\_my\_dataset.py -c best\_ckpt.pth --output-name yolox.trt

# 10. Troubleshooting

Common issues and solutions:

- \*\*CUDA Error\*\*: Reinstall torch/torchvision with the correct CUDA version.

- \*\*COCODataset has no attribute 'cache'\*\*: Modify `\_\_del\_\_` method in `datasets\_wrapper.py`.

- \*\*IndexError in Visualize\*\*: Happens if your dataset class indices exceed the default list. Use dynamic mapping.

-\*\*File\_name Error\*\* : If you encounter any error regarding “file\_name” then check your json file if it contains any key named with “file\_name”. If not then take the basename of COCO URL and replace it in yolox/tools/demo.py file.

# 11. Output Example

A typical training log includes metrics like Total Loss, IOU Loss, Class Loss, AP@0.5, and weights saved at each epoch in `YOLOX\_outputs/exp\_my\_dataset`.

You may see warnings if the number of classes doesn't match the checkpoint. Ensure correct number is set in `exp\_my\_dataset.py`.

# 12. References

- [YOLOX/docs/train\_custom\_data.md at main · Megvii-BaseDetection/YOLOX](https://github.com/Megvii-BaseDetection/YOLOX/blob/main/docs/train_custom_data.md)

- [YOLOX/README.md at main · Megvii-BaseDetection/YOLOX](https://github.com/Megvii-BaseDetection/YOLOX/blob/main/README.md)

- [YOLOX Object Detector Paper Explanation and Custom Training | LearnOpenCV #](https://learnopencv.com/yolox-object-detector-paper-explanation-and-custom-training/)

- [Faster than GPU: How to 10x your Object Detection Model and Deploy on CPU at 50+ FPS](https://dicksonneoh.com/portfolio/how_to_10x_your_od_model_and_deploy_50fps_cpu/)

- [Train Custom Data — YOLOX 0.2.0 documentation](https://yolox.readthedocs.io/en/latest/train_custom_data.html#train)