ABANDONED OBJECT DETECTION USING YOLOX

## Best Practice

If you're doing training and inference using custom datasets (like LVIS), you should avoid importing COCO\_CLASSES or VOC\_CLASSES anywhere.

You can also delete or ignore coco\_classes.py entirely if you're not working with COCO.

***Steps to follow for Training Custom Dataset using YOLOX-S :-***

1. Create a New Environment Using Conda

conda create --name yolox python=3.10 -y

conda activate yolox

IMP –

Install Dependencies

FOR GPU – ( GPU is necessary and without it, many errors will occur )

Check the **requirements.txt** file that you have cloned in the YOLOX GitHub repository. It has **torch** and **torchvision** which requires CUDA to be set up globally beforehand. If you haven’t done it before, it will install the CPU versions. This is for utilizing the GPU. You may otherwise skip this. ( CUDA 12.6 )

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

We also faced conflicts with **onnx**while installing. Hence we will skip the installation as it is not needed for training. If needed later, you can install them by removing specific versions. Now go ahead and comment out the following in the **requirements.txt** file. Don’t forget to save the changes.

IF NOT GPU -

**# torch>=1.7**

**# torchvision**

**# onnx==1.8.1**

**# onnxruntime==1.8.0**

**# onnx-simplifier==0.3.5**

Install the rest of the dependencies using the following command.

pip install -e .

OR

pip install -v -e .

2. **Prepare Your Dataset**

Since your dataset is in PASCAL VOC format, and YOLOX requires COCO format, you'll need to convert your annotations.

( best option is to download the dataset in the COCO JSON format only )

**a. Convert PASCAL VOC to COCO Format**

Use the voc2coco.py script from the yukkyo/voc2coco repository to perform the conversion.​

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

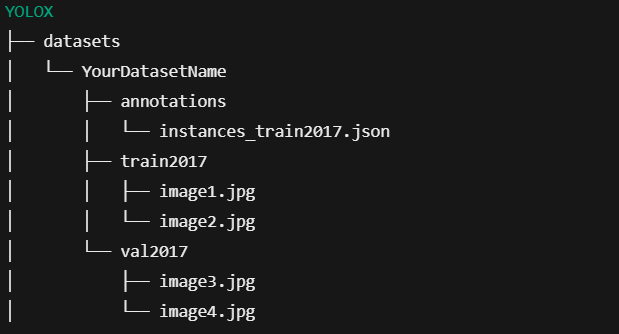
**cd voc2coco**

**python voc2coco.py --ann\_dir path\_to\_voc\_annotations --ann\_ids path\_to\_image\_ids --labels path\_to\_labels --output path\_to\_output\_json**

Replace path\_to\_voc\_annotations, path\_to\_image\_ids, path\_to\_labels, and path\_to\_output\_json with your actual paths.

**b. Organize the Dataset Directory Structure**

After conversion, structure your dataset as follows:

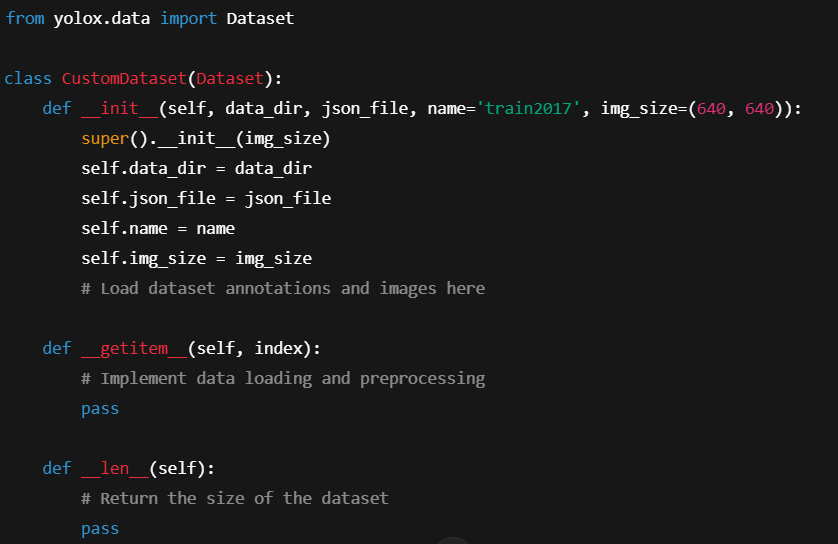


Place your COCO-format annotation file in the annotations directory and your training images in the train2017 directory. If you have validation images, place them in the val2017 directory.

**3. Implement a Custom Dataset Class**

YOLOX provides built-in support for COCO datasets. However, if your dataset has specific requirements, you may need to implement a custom dataset class.





This class should handle loading images and annotations, applying transformations, and returning data in the format expected by YOLOX.

This file contains the COCODataset class, which is designed to handle datasets in COCO format. Therefore, you do not need to create a new dataset class; you can use the existing COCODataset class by specifying the appropriate paths in your experiment configuration file.

**4. Configure the YOLOX Experiment**

**A ) Navigate to the exps/default directory:**

A black and white text

AI-generated content may be incorrect.

**B ) Copy the yolox\_s.py file to create a new configuration file for your dataset:**

A black and white text

AI-generated content may be incorrect.

Edit exp\_my\_dataset.py to specify your dataset paths and any other parameters specific to your dataset. Since you only have a training dataset (train2017), ensure that the validation dataset paths are handled appropriately, possibly by setting them to None or an empty list.

**Modify the \_\_init\_\_ method**

Update your exp\_my\_dataset.py file like this:

**#!/usr/bin/env python3**

**# -\*- coding:utf-8 -\*-**

**# Copyright (c) Megvii, Inc. and its affiliates.**

**import os**

**from yolox.exp import Exp as MyExp**

**from yolox.data.datasets import COCODataset # Import COCO dataset handler**

**class Exp(MyExp):**

**def \_\_init\_\_(self):**

**super(Exp, self).\_\_init\_\_()**

**# Model width and depth (you can change based on YOLOX versions: s, m, l, etc.)**

**self.depth = 0.33 # Controls model depth**

**self.width = 0.50 # Controls model width**

**# Experiment name**

**self.exp\_name = os.path.split(os.path.realpath(\_\_file\_\_))[1].split(".")[0]**

**# Modify dataset paths**

**self.data\_dir = "/path/to/your/dataset" # Change this to your actual dataset location**

**self.train\_ann = "annotations/instances\_train2017.json" # COCO format annotation file**

**self.val\_ann = None # No validation set, you can set this to None**

**self.test\_ann = None # No test set**

**self.num\_classes= 14 #set number of classes**

**# Adjust batch size (depends on your GPU memory)**

**self.batch\_size = 8**

**self.num\_workers = 4 # Number of data loader workers**

**# Maximum number of training epochs**

**self.max\_epoch = 50 # You can adjust based on dataset size**

**# Input image size**

**self.input\_size = (640, 640) # Adjust if necessary**

**# Set evaluation intervals (optional, since you don't have a validation dataset)**

**self.eval\_interval = 1 # Evaluate every epoch (set to higher if training takes too long)**

(Note:- path to dataset directory means the folder that contains both annotations and train folders)

**5. Train the Model**

Run the training script with your custom experiment configuration.

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

Adjust the -d (number of devices) and -b (batch size) parameters based on your hardware capabilities.

**6. Evaluate the Model**

After training, evaluate your model's performance.

(Note: the below command is in full one line)

**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**

This command uses the best checkpoint saved during training to evaluate the model on the validation set.

**RESOURCES :-**

[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" \l "train)

**Next Steps to Use Your Trained Model**

**1. Evaluate the Model**

Check its performance on your validation/test set:

Bash

**python tools/eval.py -f exps/your\_exp.py -c YOLOX\_outputs/exp\_name/best\_ckpt.pth -b 8 -d 1 --conf 0.01 [--fp16] [--fuse]**

* --conf: Adjust confidence threshold (default: 0.01)
* --fp16: Enable mixed-precision (if supported)
* --fuse: Fuse Conv+BN for faster inference

**2. Run Inference on Images/Videos**

Use the trained model for detection:

For single image:

bash

**python tools/demo.py image -f exps/your\_exp.py -c YOLOX\_outputs/exp\_name/best\_ckpt.pth --path assets/dog.jpg --conf 0.3 --nms 0.65 --tsize 640 --save\_result --device [gpu|cpu]**

For webcam/video:

bash

**python tools/demo.py webcam -f exps/your\_exp.py -c YOLOX\_outputs/exp\_name/best\_ckpt.pth --conf 0.3 --nms 0.65 --tsize 640 --device gpu**

* --tsize: Input size (should match training, e.g., 640)

**3. Analyze Training Metrics**

* TensorBoard (visualize loss, mAP, LR curves):

bash

**tensorboard --logdir YOLOX\_outputs/exp\_name**

Open http://localhost:6006 in your browser.

* Check train\_log.txt for:
  + Epoch-wise AP@0.5:0.95, AP@0.5
  + Loss trends (total\_loss, iou\_loss, cls\_loss)

**4. Export to ONNX/TensorRT (Optional)**

**Convert to ONNX for deployment:**

**bash**

**Copy**

**python tools/export\_onnx.py -f exps/your\_exp.py -c YOLOX\_outputs/exp\_name/best\_ckpt.pth --output-name yolox.onnx**

**For TensorRT:**

**bash**

**Copy**

**python tools/trt.py -f exps/your\_exp.py -c best\_ckpt.pth --output-name yolox.trt**

**5. Fine-Tune (If Needed)**

**If performance is poor:**

1. **Increase training epochs: Adjust self.max\_epoch in exp.py.**
2. **Data Augmentation: Modify self.mosaic\_prob, self.mixup\_prob in exp.py.**
3. **Hyperparameters: Tune LR (self.basic\_lr\_per\_img), optimizer (SGD/AdamW).**

**Key Files & Their Uses**

| **File/Folder** | **Purpose** |
| --- | --- |
| **best\_ckpt.pth** | **Best weights (for deployment)** |
| **latest\_ckpt.pth** | **Last checkpoint (resume training)** |
| **train\_log.txt** | **Debug training issues (check loss/AP)** |
| **TensorBoard logs** | **Visualize metrics (e.g., AP, loss\_l1, lr)** |
| **vis\_train folder** | **Training batch visualizations (verify data loading)** |

**ERRORS :-**

1. File "C:\Users\patel\anaconda3\envs\yolox\lib\subprocess.py", line 526, in run raise CalledProcessError(retcode, process.args, │ │ │ └ ['where', 'cl'] │ │ └ <Popen: returncode: 1 args: ['where', 'cl']> │ └ 1 └ <class 'subprocess.CalledProcessError'> subprocess.CalledProcessError: Command '['where', 'cl']' returned non-zero exit status   
     
   Solution :- Check in exp file if there is a line written as – “self.fast\_eval = false”. If yes then remove that line. Clear the Cache from c:/users/<user\_name>/app\_data/local/torch\_extensions/cache (delete this cache file). Also check if MSVC build tools are present or not. If not then download it. Then ensure required dependencies are installed using – “**pip install -r requirements.txt**” .
2. If any CUDA related error occurs then uninstall the torch torchvision and torchaudio and reinstall it again from pytorch official website.
3. ‘COCODateset’ has no attribute ‘cache’ – if this error occurs then modify this -   
   his error occurs when the \_\_del\_\_ method in your CacheDataset class is called on an object that wasn't fully initialized, possibly due to a prior failure during dataset loading (such as the 'iscrowd' key error). Because initialization failed, self.cache was never defined, but the destructor (\_\_del\_\_) is still called, resulting in the error.

#### ✅ Fix

Update the \_\_del\_\_ method in datasets\_wrapper.py like this to avoid errors when attributes don't exist:

python

CopyEdit

def \_\_del\_\_(self):

if hasattr(self, "cache") and self.cache and getattr(self, "cache\_type", None) == "ram":

if hasattr(self, "imgs"):

del self.imgs

1. cls\_ids :: tensor([1018., 1025., 206., 206., 206., 1025.])

scores :: tensor([0.5827, 0.1228, 0.0565, 0.0402, 0.0194, 0.0130])

conf :: 0.01

class\_names :: ('person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush')

Traceback (most recent call last):

File "/home/tuskerai/Desktop/Dev Patel/YOLOX/tools/demo.py", line 320, in <module>

main(exp, args)

File "/home/tuskerai/Desktop/Dev Patel/YOLOX/tools/demo.py", line 311, in main

image\_demo(predictor, vis\_folder, args.path, current\_time, args.save\_result)

File "/home/tuskerai/Desktop/Dev Patel/YOLOX/tools/demo.py", line 195, in image\_demo

result\_image = predictor.visual(outputs[0], img\_info, predictor.confthre)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "/home/tuskerai/Desktop/Dev Patel/YOLOX/tools/demo.py", line 183, in visual

vis\_res = vis(img, bboxes, scores, cls, cls\_conf, self.cls\_names)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "/home/tuskerai/Desktop/Dev Patel/YOLOX/yolox/utils/visualize.py", line 28, in vis

color = (\_COLORS[cls\_id] \* 255).astype(np.uint8).tolist()

~~~~~~~^^^^^^^^

IndexError: index 1018 is out of bounds for axis 0 with size 80  
  
SOLUTION – this happens because of the default class names specified in coco\_classes.py file and the classes in your annotations file. I am adding a dynamic command line argument where we could specify the json file path so that model can read that json file class name.

**OUTPUT :-**

(yolox) D:\Projects\abandoned\YOLOX>python tools/train.py -f exps/default/exp\_my\_dataset.py -d 1 -b 8 --fp16 -c yolox\_s.pth

\* History restored

Microsoft Windows [Version 10.0.26100.3476]

(c) Microsoft Corporation. All rights reserved.

(yolox) D:\Projects\abandoned>cd yolox

(yolox) D:\Projects\abandoned\YOLOX>python tools/train.py -f exps/default/exp\_my\_dataset.py -d 1 -b 8 --fp16 -c yolox\_s.pth

d:\projects\abandoned\yolox\yolox\core\trainer.py:47: FutureWarning: `torch.cuda.amp.GradScaler(args...)` is deprecated. Please use `torch.amp.GradScaler('cuda', args...)` instead.

self.scaler = torch.cuda.amp.GradScaler(enabled=args.fp16)

2025-04-01 17:46:19 | INFO | yolox.core.trainer:132 - args: Namespace(experiment\_name='exp\_my\_dataset', name=None, dist\_backend='nccl', dist\_url=None, batch\_size=8, devices=1, exp\_file='exps/default/exp\_my\_dataset.py', resume=False, ckpt='yolox\_s.pth', start\_epoch=None, num\_machines=1, machine\_rank=0, fp16=True, cache=None, occupy=False, logger='tensorboard', opts=[])

2025-04-01 17:46:19 | INFO | yolox.core.trainer:133 - exp value:

╒═══════════════════╤═════════════════════════════════════════════════════════════════════════════════════════════╕

│ keys │ values │

╞═══════════════════╪═════════════════════════════════════════════════════════════════════════════════════════════╡

│ seed │ None │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ output\_dir │ './YOLOX\_outputs' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ print\_interval │ 10 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ eval\_interval │ 1 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ dataset │ None │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ num\_classes │ 14 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ depth │ 0.33 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ width │ 0.5 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ act │ 'silu' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ data\_num\_workers │ 4 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ input\_size │ (640, 640) │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ multiscale\_range │ 5 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ data\_dir │ 'D:\\Projects\\abandoned\\YOLOX\\datasets\\chiikawa' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ train\_ann │ 'D:\\Projects\\abandoned\\YOLOX\\datasets\\chiikawa\\annotations\\instances\_train2017.json' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ val\_ann │ 'D:\\Projects\\abandoned\\YOLOX\\datasets\\chiikawa\\annotations\\instances\_val2017.json' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ test\_ann │ 'instances\_test2017.json' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ mosaic\_prob │ 1.0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ mixup\_prob │ 1.0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ hsv\_prob │ 1.0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ flip\_prob │ 0.5 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ degrees │ 10.0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ translate │ 0.1 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ mosaic\_scale │ (0.1, 2) │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ enable\_mixup │ True │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ mixup\_scale │ (0.5, 1.5) │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ shear │ 2.0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ warmup\_epochs │ 5 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ max\_epoch │ 2 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ warmup\_lr │ 0 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ min\_lr\_ratio │ 0.05 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ basic\_lr\_per\_img │ 0.00015625 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ scheduler │ 'yoloxwarmcos' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ no\_aug\_epochs │ 15 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ ema │ True │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ weight\_decay │ 0.0005 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ momentum │ 0.9 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ save\_history\_ckpt │ True │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ exp\_name │ 'exp\_my\_dataset' │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ test\_size │ (640, 640) │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ test\_conf │ 0.01 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ nmsthre │ 0.65 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ batch\_size │ 8 │

├───────────────────┼─────────────────────────────────────────────────────────────────────────────────────────────┤

│ num\_workers │ 4 │

╘═══════════════════╧═════════════════════════════════════════════════════════════════════════════════════════════╛

2025-04-01 17:46:20 | INFO | yolox.core.trainer:138 - Model Summary: Params: 8.94M, Gflops: 26.78

2025-04-01 17:46:21 | INFO | yolox.core.trainer:338 - loading checkpoint for fine tuning

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.0.weight in checkpoint is torch.Size([80, 128, 1, 1]), while shape of head.cls\_preds.0.weight in model is torch.Size([14, 128, 1, 1]).

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.0.bias in checkpoint is torch.Size([80]), while shape of head.cls\_preds.0.bias in model is torch.Size([14]).

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.1.weight in checkpoint is torch.Size([80, 128, 1, 1]), while shape of head.cls\_preds.1.weight in model is torch.Size([14, 128, 1, 1]).

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.1.bias in checkpoint is torch.Size([80]), while shape of head.cls\_preds.1.bias in model is torch.Size([14]).

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.2.weight in checkpoint is torch.Size([80, 128, 1, 1]), while shape of head.cls\_preds.2.weight in model is torch.Size([14, 128, 1, 1]).

2025-04-01 17:46:24 | WARNING | yolox.utils.checkpoint:24 - Shape of head.cls\_preds.2.bias in checkpoint is torch.Size([80]), while shape of head.cls\_preds.2.bias in model is torch.Size([14]).

2025-04-01 17:46:24 | INFO | yolox.data.datasets.coco:63 - loading annotations into memory...

2025-04-01 17:46:24 | INFO | yolox.data.datasets.coco:63 - Done (t=0.03s)

2025-04-01 17:46:24 | INFO | pycocotools.coco:86 - creating index...

2025-04-01 17:46:24 | INFO | pycocotools.coco:86 - index created!

2025-04-01 17:46:24 | INFO | yolox.core.trainer:157 - init prefetcher, this might take one minute or less...

2025-04-01 17:47:00 | INFO | yolox.data.datasets.coco:63 - loading annotations into memory...

2025-04-01 17:47:01 | INFO | yolox.data.datasets.coco:63 - Done (t=0.04s)

2025-04-01 17:47:01 | INFO | pycocotools.coco:86 - creating index...

2025-04-01 17:47:01 | INFO | pycocotools.coco:86 - index created!

2025-04-01 17:47:01 | INFO | yolox.core.trainer:196 - Training start...

2025-04-01 17:47:01 | INFO | yolox.core.trainer:197 -

YOLOX(

(backbone): YOLOPAFPN(

(backbone): CSPDarknet(

(stem): Focus(

(conv): BaseConv(

(conv): Conv2d(12, 32, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(dark2): Sequential(

(0): BaseConv(

(conv): Conv2d(32, 64, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(64, 32, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(64, 32, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(32, 32, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(32, 32, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

)

(dark3): Sequential(

(0): BaseConv(

(conv): Conv2d(64, 128, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(128, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(1): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(2): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

)

(dark4): Sequential(

(0): BaseConv(

(conv): Conv2d(128, 256, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(1): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(2): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

)

(dark5): Sequential(

(0): BaseConv(

(conv): Conv2d(256, 512, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(512, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): SPPBottleneck(

(conv1): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): ModuleList(

(0): MaxPool2d(kernel\_size=5, stride=1, padding=2, dilation=1, ceil\_mode=False)

(1): MaxPool2d(kernel\_size=9, stride=1, padding=4, dilation=1, ceil\_mode=False)

(2): MaxPool2d(kernel\_size=13, stride=1, padding=6, dilation=1, ceil\_mode=False)

)

(conv2): BaseConv(

(conv): Conv2d(1024, 512, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(512, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(2): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(512, 512, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(512, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

)

)

(upsample): Upsample(scale\_factor=2.0, mode='nearest')

(lateral\_conv0): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(C3\_p4): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(512, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(512, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

(reduce\_conv1): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(C3\_p3): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(256, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(256, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

(bu\_conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(C3\_n3): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

(bu\_conv1): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(C3\_n4): CSPLayer(

(conv1): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(512, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv3): BaseConv(

(conv): Conv2d(512, 512, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(512, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(m): Sequential(

(0): Bottleneck(

(conv1): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(conv2): BaseConv(

(conv): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

)

)

(head): YOLOXHead(

(cls\_convs): ModuleList(

(0-2): 3 x Sequential(

(0): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

(reg\_convs): ModuleList(

(0-2): 3 x Sequential(

(0): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

)

(cls\_preds): ModuleList(

(0-2): 3 x Conv2d(128, 14, kernel\_size=(1, 1), stride=(1, 1))

)

(reg\_preds): ModuleList(

(0-2): 3 x Conv2d(128, 4, kernel\_size=(1, 1), stride=(1, 1))

)

(obj\_preds): ModuleList(

(0-2): 3 x Conv2d(128, 1, kernel\_size=(1, 1), stride=(1, 1))

)

(stems): ModuleList(

(0): BaseConv(

(conv): Conv2d(128, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(1): BaseConv(

(conv): Conv2d(256, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

(2): BaseConv(

(conv): Conv2d(512, 128, kernel\_size=(1, 1), stride=(1, 1), bias=False)

(bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track\_running\_stats=True)

(act): SiLU(inplace=True)

)

)

(l1\_loss): L1Loss()

(bcewithlog\_loss): BCEWithLogitsLoss()

(iou\_loss): IOUloss()

)

)

2025-04-01 17:47:01 | INFO | yolox.core.trainer:218 - ---> start train epoch1

2025-04-01 17:47:01 | INFO | yolox.core.trainer:221 - --->No mosaic aug now!

2025-04-01 17:47:01 | INFO | yolox.core.trainer:223 - --->Add additional L1 loss now!

d:\projects\abandoned\yolox\yolox\core\trainer.py:106: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.

with torch.cuda.amp.autocast(enabled=self.amp\_training):

d:\projects\abandoned\yolox\yolox\models\yolo\_head.py:474: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.

with torch.cuda.amp.autocast(enabled=False):

2025-04-01 17:47:10 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 10/61, gpu mem: 1729Mb, mem: 10.9Gb, iter\_time: 0.893s, data\_time: 0.005s, total\_loss: 10.2, iou\_loss: 2.1, l1\_loss: 1.0, conf\_loss: 3.8, cls\_loss: 3.3, lr: 1.344e-06, size: 640, ETA: 0:01:39

2025-04-01 17:47:14 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 20/61, gpu mem: 1740Mb, mem: 10.9Gb, iter\_time: 0.423s, data\_time: 0.001s, total\_loss: 11.1, iou\_loss: 2.3, l1\_loss: 1.1, conf\_loss: 4.1, cls\_loss: 3.6, lr: 5.375e-06, size: 640, ETA: 0:01:07

2025-04-01 17:47:20 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 30/61, gpu mem: 1740Mb, mem: 10.9Gb, iter\_time: 0.610s, data\_time: 0.005s, total\_loss: 9.6, iou\_loss: 1.6, l1\_loss: 0.7, conf\_loss: 3.6, cls\_loss: 3.7, lr: 1.209e-05, size: 576, ETA: 0:00:59

2025-04-01 17:47:28 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 40/61, gpu mem: 1740Mb, mem: 10.9Gb, iter\_time: 0.835s, data\_time: 0.004s, total\_loss: 9.4, iou\_loss: 1.5, l1\_loss: 0.8, conf\_loss: 3.3, cls\_loss: 3.9, lr: 2.150e-05, size: 480, ETA: 0:00:56

2025-04-01 17:47:52 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 50/61, gpu mem: 2421Mb, mem: 10.8Gb, iter\_time: 2.333s, data\_time: 0.002s, total\_loss: 7.9, iou\_loss: 1.5, l1\_loss: 1.1, conf\_loss: 2.7, cls\_loss: 2.6, lr: 3.359e-05, size: 768, ETA: 0:01:13

2025-04-01 17:47:59 | INFO | yolox.core.trainer:270 - epoch: 1/2, iter: 60/61, gpu mem: 2421Mb, mem: 10.7Gb, iter\_time: 0.716s, data\_time: 0.003s, total\_loss: 8.2, iou\_loss: 1.8, l1\_loss: 0.8, conf\_loss: 2.7, cls\_loss: 2.8, lr: 4.837e-05, size: 480, ETA: 0:01:00

2025-04-01 17:48:07 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

100%|####################################################################################################################################| 61/61 [01:13<00:00, 1.20s/it]

d:\projects\abandoned\yolox\yolox\evaluators\coco\_evaluator.py:189: UserWarning: The torch.cuda.\*DtypeTensor constructors are no longer recommended. It's best to use methods such as torch.tensor(data, dtype=\*, device='cuda') to create tensors. (Triggered internally at C:\actions-runner\\_work\pytorch\pytorch\pytorch\torch\csrc\tensor\python\_tensor.cpp:80.)

statistics = torch.cuda.FloatTensor([inference\_time, nms\_time, n\_samples])

2025-04-01 17:49:21 | INFO | yolox.evaluators.coco\_evaluator:259 - Evaluate in main process...

2025-04-01 17:49:22 | INFO | yolox.evaluators.coco\_evaluator:292 - Loading and preparing results...

2025-04-01 17:49:22 | INFO | yolox.evaluators.coco\_evaluator:292 - DONE (t=0.13s)

2025-04-01 17:49:22 | INFO | pycocotools.coco:366 - creating index...

2025-04-01 17:49:22 | INFO | pycocotools.coco:366 - index created!

Using C:\Users\patel\AppData\Local\torch\_extensions\torch\_extensions\Cache\py310\_cu126 as PyTorch extensions root...

Creating extension directory C:\Users\patel\AppData\Local\torch\_extensions\torch\_extensions\Cache\py310\_cu126\fast\_cocoeval...

Emitting ninja build file C:\Users\patel\AppData\Local\torch\_extensions\torch\_extensions\Cache\py310\_cu126\fast\_cocoeval\build.ninja...

Building extension module fast\_cocoeval...

Allowing ninja to set a default number of workers... (overridable by setting the environment variable MAX\_JOBS=N)

[1/2] cl /showIncludes -DTORCH\_EXTENSION\_NAME=fast\_cocoeval -DTORCH\_API\_INCLUDE\_EXTENSION\_H -IC:\Users\patel\anaconda3\envs\yolox\lib\site-packages\torch\include -IC:\Users\patel\anaconda3\envs\yolox\lib\site-packages\torch\include\torch\csrc\api\include -IC:\Users\patel\anaconda3\envs\yolox\lib\site-packages\torch\include\TH -IC:\Users\patel\anaconda3\envs\yolox\lib\site-packages\torch\include\THC -IC:\Users\patel\anaconda3\envs\yolox\Include -D\_GLIBCXX\_USE\_CXX11\_ABI=0 /MD /wd4819 /wd4251 /wd4244 /wd4267 /wd4275 /wd4018 /wd4190 /wd4624 /wd4067 /wd4068 /EHsc /std:c++17 -O2 -c D:\Projects\abandoned\YOLOX\yolox\layers\cocoeval\cocoeval.cpp /Fococoeval.o

Microsoft (R) C/C++ Optimizing Compiler Version 19.43.34808 for x64

Copyright (C) Microsoft Corporation. All rights reserved.

[2/2] "C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\Tools\MSVC\14.43.34808\bin\Hostx64\x64/link.exe" cocoeval.o /nologo /DLL c10.lib torch\_cpu.lib torch.lib /LIBPATH:C:\Users\patel\anaconda3\envs\yolox\lib\site-packages\torch\lib torch\_python.lib /LIBPATH:C:\Users\patel\anaconda3\envs\yolox\libs /out:fast\_cocoeval.pyd

Creating library fast\_cocoeval.lib and object fast\_cocoeval.exp

Loading extension module fast\_cocoeval...

Running per image evaluation...

2025-04-01 17:49:39 | INFO | yolox.layers.jit\_ops:111 - Load fast\_cocoeval op in 15.696s.

Evaluate annotation type \*bbox\*

COCOeval\_opt.evaluate() finished in 0.44 seconds.

Accumulating evaluation results...

COCOeval\_opt.accumulate() finished in 0.09 seconds.

2025-04-01 17:49:40 | INFO | yolox.core.trainer:381 -

Average forward time: 61.11 ms, Average NMS time: 2.93 ms, Average inference time: 64.04 ms

Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.041

Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.070

Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.040

Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.034

Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.044

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.092

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.187

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.222

Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.162

Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.243

per class AP:

| class | AP | class | AP | class | AP |

|:---------|:-------|:--------|:-------|:----------|:------|

| doll2 | 0.249 | 3dolls | 0.000 | Sea\_otter | 1.252 |

| Shisa | 1.531 | anoko | 10.321 | beetle | 0.000 |

| chiikawa | 10.786 | fish | 0.620 | kani | 0.000 |

| kurima | 1.687 | momo | 2.687 | small ba | 7.110 |

| usaki | 16.552 | | | | |

per class AR:

| class | AR | class | AR | class | AR |

|:---------|:-------|:--------|:-------|:----------|:-------|

| doll2 | 5.455 | 3dolls | 0.000 | Sea\_otter | 12.881 |

| Shisa | 28.256 | anoko | 30.455 | beetle | 0.000 |

| chiikawa | 56.806 | fish | 10.000 | kani | 0.000 |

| kurima | 21.526 | momo | 25.381 | small ba | 35.579 |

| usaki | 61.758 | | | | |

2025-04-01 17:49:40 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

2025-04-01 17:49:41 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

2025-04-01 17:49:41 | INFO | yolox.core.trainer:218 - ---> start train epoch2

2025-04-01 17:49:41 | INFO | yolox.core.trainer:221 - --->No mosaic aug now!

2025-04-01 17:49:41 | INFO | yolox.core.trainer:223 - --->Add additional L1 loss now!

2025-04-01 17:49:53 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 10/61, gpu mem: 2421Mb, mem: 11.4Gb, iter\_time: 1.133s, data\_time: 0.002s, total\_loss: 8.2, iou\_loss: 1.5, l1\_loss: 0.8, conf\_loss: 3.0, cls\_loss: 2.8, lr: 6.774e-05, size: 576, ETA: 0:00:55

2025-04-01 17:50:10 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 20/61, gpu mem: 2421Mb, mem: 11.3Gb, iter\_time: 1.763s, data\_time: 0.003s, total\_loss: 7.0, iou\_loss: 1.7, l1\_loss: 1.1, conf\_loss: 2.4, cls\_loss: 1.8, lr: 8.816e-05, size: 704, ETA: 0:00:48

2025-04-01 17:50:27 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 30/61, gpu mem: 2421Mb, mem: 11.3Gb, iter\_time: 1.686s, data\_time: 0.004s, total\_loss: 8.3, iou\_loss: 1.9, l1\_loss: 1.2, conf\_loss: 3.1, cls\_loss: 2.1, lr: 1.113e-04, size: 512, ETA: 0:00:38

2025-04-01 17:50:35 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 40/61, gpu mem: 2421Mb, mem: 11.4Gb, iter\_time: 0.807s, data\_time: 0.003s, total\_loss: 6.9, iou\_loss: 1.5, l1\_loss: 0.8, conf\_loss: 2.5, cls\_loss: 2.0, lr: 1.371e-04, size: 672, ETA: 0:00:25

2025-04-01 17:50:46 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 50/61, gpu mem: 2421Mb, mem: 11.4Gb, iter\_time: 1.100s, data\_time: 0.004s, total\_loss: 7.9, iou\_loss: 1.7, l1\_loss: 1.0, conf\_loss: 2.6, cls\_loss: 2.6, lr: 1.656e-04, size: 512, ETA: 0:00:13

2025-04-01 17:50:55 | INFO | yolox.core.trainer:270 - epoch: 2/2, iter: 60/61, gpu mem: 2421Mb, mem: 11.4Gb, iter\_time: 0.847s, data\_time: 0.003s, total\_loss: 7.1, iou\_loss: 1.7, l1\_loss: 1.0, conf\_loss: 2.9, cls\_loss: 1.6, lr: 1.967e-04, size: 768, ETA: 0:00:01

2025-04-01 17:50:56 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

100%|####################################################################################################################################| 61/61 [01:06<00:00, 1.10s/it]

2025-04-01 17:52:03 | INFO | yolox.evaluators.coco\_evaluator:259 - Evaluate in main process...

2025-04-01 17:52:04 | INFO | yolox.evaluators.coco\_evaluator:292 - Loading and preparing results...

2025-04-01 17:52:04 | INFO | yolox.evaluators.coco\_evaluator:292 - DONE (t=0.13s)

2025-04-01 17:52:04 | INFO | pycocotools.coco:366 - creating index...

2025-04-01 17:52:04 | INFO | pycocotools.coco:366 - index created!

Using C:\Users\patel\AppData\Local\torch\_extensions\torch\_extensions\Cache\py310\_cu126 as PyTorch extensions root...

No modifications detected for re-loaded extension module fast\_cocoeval, skipping build step...

Loading extension module fast\_cocoeval...

Running per image evaluation...

2025-04-01 17:52:04 | INFO | yolox.layers.jit\_ops:111 - Load fast\_cocoeval op in 0.011s.

Evaluate annotation type \*bbox\*

COCOeval\_opt.evaluate() finished in 0.51 seconds.

Accumulating evaluation results...

COCOeval\_opt.accumulate() finished in 0.07 seconds.

2025-04-01 17:52:05 | INFO | yolox.core.trainer:381 -

Average forward time: 54.04 ms, Average NMS time: 2.59 ms, Average inference time: 56.63 ms

Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.121

Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.192

Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.138

Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.077

Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.121

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.155

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.241

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.264

Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.160

Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.277

per class AP:

| class | AP | class | AP | class | AP |

|:---------|:-------|:--------|:-------|:----------|:-------|

| doll2 | 0.000 | 3dolls | 0.891 | Sea\_otter | 0.321 |

| Shisa | 0.304 | anoko | 31.492 | beetle | 0.000 |

| chiikawa | 12.115 | fish | 58.214 | kani | 0.000 |

| kurima | 2.037 | momo | 8.044 | small ba | 10.675 |

| usaki | 33.144 | | | | |

per class AR:

| class | AR | class | AR | class | AR |

|:---------|:-------|:--------|:-------|:----------|:-------|

| doll2 | 0.000 | 3dolls | 2.500 | Sea\_otter | 9.492 |

| Shisa | 5.814 | anoko | 46.818 | beetle | 0.000 |

| chiikawa | 56.060 | fish | 81.667 | kani | 0.000 |

| kurima | 21.526 | momo | 30.355 | small ba | 28.813 |

| usaki | 60.546 | | | | |

2025-04-01 17:52:05 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

2025-04-01 17:52:05 | INFO | yolox.core.trainer:402 - Save weights to ./YOLOX\_outputs\exp\_my\_dataset

2025-04-01 17:52:06 | INFO | yolox.core.trainer:200 - Training of experiment is done and the best AP is 12.10

**CHANGES I DID:-**

1. FOR CLASS\_NAMES ERROR WHEN USING LVIS DATASET

### Step 1: Remove this line (in demo.py file)

python

CopyEdit

from yolox.data.datasets import COCO\_CLASSES

### ✅ Step 2: Add LVIS class name loader

Place this near the top of your script:

python

CopyEdit

import json

def load\_lvis\_class\_name\_list(json\_path, num\_classes):

with open(json\_path, 'r') as f:

data = json.load(f)

# Sort by ID assuming remapped 0..1202

categories = sorted(data['categories'], key=lambda x: x['id'])

class\_names = [cat['name'] for cat in categories]

# Ensure the list has exactly num\_classes

assert len(class\_names) == num\_classes, f"Expected {num\_classes} classes, got {len(class\_names)}"

return class\_names

### ✅ Step 3: Replace this block in main():

python

CopyEdit

predictor = Predictor(

model, exp, COCO\_CLASSES, trt\_file, decoder,

args.device, args.fp16, args.legacy,

)

#### With this:

python

CopyEdit

# Provide the path to your LVIS annotation file

lvis\_ann\_path = "/home/tuskerai/Desktop/Dev Patel/YOLOX/datasets/abandoned/annotations/instances\_val2017.json"

# Load class names dynamically from the annotation

cls\_names = load\_lvis\_class\_name\_list(lvis\_ann\_path, exp.num\_classes)

predictor = Predictor(

model, exp, cls\_names, trt\_file, decoder,

args.device, args.fp16, args.legacy,

)

Great idea — you're aiming for a flexible and clean command-line interface, where:

* ✅ If the user provides a --class\_json path, it uses that to load class names.
* ✅ If the user doesn't provide anything, it falls back to a default path (e.g. the hardcoded one in your script).

Let’s implement this in 3 steps:

## ✅ Step 1: Add --class\_json argument to make\_parser()

Find your make\_parser() function and add this:

python

CopyEdit

parser.add\_argument(

"--class\_json",

type=str,

default=None,

help="Optional: path to LVIS-style annotation JSON for loading class names"

)

## ✅ Step 2: Modify main() function to handle this logic

Replace this block in your main():

python

CopyEdit

# Existing COCO\_CLASSES usage — REMOVE THIS:

# predictor = Predictor(

# model, exp, COCO\_CLASSES, trt\_file, decoder,

# args.device, args.fp16, args.legacy,

# )

With this logic:

python

CopyEdit

# Fallback default path if --class\_json is not provided

DEFAULT\_LVIS\_JSON = "/home/tuskerai/Desktop/Dev Patel/YOLOX/datasets/abandoned/annotations/instances\_val2017.json"

if args.class\_json:

class\_json\_path = args.class\_json

else:

class\_json\_path = DEFAULT\_LVIS\_JSON

cls\_names = load\_lvis\_class\_name\_list(class\_json\_path, exp.num\_classes)

predictor = Predictor(

model, exp, cls\_names, trt\_file, decoder,

args.device, args.fp16, args.legacy,

)

2. ❌ Problem

You're using a fixed \_COLORS array:

python

CopyEdit

\_COLORS = np.array([...]).reshape(-1, 3)

* It only defines 80 colors (approx).
* But your cls\_ids contains values like 206, 1177, etc.
* So this line:

python

CopyEdit

color = (\_COLORS[cls\_id] \* 255).astype(np.uint8).tolist()

throws an IndexError when cls\_id >= 80.

## ✅ Solution: Dynamically Generate Colors Based on Number of Classes

Here's how you can fix it safely and cleanly, so it works with any number of classes (including 1203 for LVIS):

### 🔁 Step 1: Replace \_COLORS definition with a dynamic color generator

At the top of vis.py, replace your \_COLORS array with this:

python

CopyEdit

import random

def generate\_colors(num\_classes):

random.seed(42) # For reproducibility

return np.array([[random.random(), random.random(), random.random()] for \_ in range(num\_classes)], dtype=np.float32)

### ✨ Step 2: Move \_COLORS inside the vis() function

Now, inside def vis(...), add:

python

CopyEdit

num\_classes = len(class\_names)

\_COLORS = generate\_colors(num\_classes)

This ensures \_COLORS always has enough entries.

### 🔧 Final vis() Function Should Look Like This:

python

CopyEdit

def vis(img, boxes, scores, cls\_ids, conf=0.5, class\_names=None):

print(f"cls\_ids :: {cls\_ids}")

print(f"scores :: {scores}")

print(f"conf :: {conf}")

print(f"class\_names :: {class\_names}")

num\_classes = len(class\_names)

\_COLORS = generate\_colors(num\_classes)

for i in range(len(boxes)):

box = boxes[i]

cls\_id = int(cls\_ids[i])

score = scores[i]

if score < conf:

continue

x0 = int(box[0])

y0 = int(box[1])

x1 = int(box[2])

y1 = int(box[3])

color = (\_COLORS[cls\_id] \* 255).astype(np.uint8).tolist()

# You already override this to yellow:

color = [127, 127, 0]

text = '{}:{:.1f}%'.format(class\_names[cls\_id], score \* 100)

txt\_color = (0, 0, 0) if np.mean(\_COLORS[cls\_id]) > 0.5 else (255, 255, 255)

font = cv2.FONT\_HERSHEY\_SIMPLEX

txt\_size = cv2.getTextSize(text, font, 0.4, 1)[0]

cv2.rectangle(img, (x0, y0), (x1, y1), color, 2)

txt\_bk\_color = (\_COLORS[cls\_id] \* 255 \* 0.7).astype(np.uint8).tolist()

cv2.rectangle(

img,

(x0, y0 + 1),

(x0 + txt\_size[0] + 1, y0 + int(1.5\*txt\_size[1])),

txt\_bk\_color,

-1

)

cv2.putText(img, text, (x0, y0 + txt\_size[1]), font, 0.4, txt\_color, thickness=1)

return img

## ✅ Now You're Safe From:

* IndexError caused by class ID > 79
* Class mismatch in class\_names
* Hardcoded assumptions about COCO class count