YOLOv3 项目

使用Conda创建新的环境

```
conda create -n yolov3 python
conda activate yolov3
```

安装环境依赖

```
 python -m pip install -r requirements.txt
```

Git Clone项目

```
git clone Practice_14_YOLOv3Detector.git
```

运行获取数据集脚本

```
cd Practice_14_YOLOv3Detector
mkdir data
cd data
. get_coco_dataset.sh
```

获取预训练模型参数

```
mkdir ../weight
cd ../weight
. download_weights.sh
```

调整参数

```
parser = argparse.ArgumentParser()
    parser.add_argument("--epochs", type=int, default=3, help="number of epochs")
    parser.add_argument("--batch_size", type=int, default=300, help="size of each image batch")
    parser.add_argument("--gradient_accumulations", type=int, default=2, help="number of gradient accums before step")
    parser.add_argument("--model_def", type=str, default="config/yolov3.cfg", help="path to model definition file")
    parser.add_argument("--data_config", type=str, default="config/coco.data", help="path to data config file")
    parser.add_argument("--pretrained_weights", type=str, help="if specified starts from checkpoint model")
    parser.add_argument("--n_cpu", type=int, default=12, help="number of cpu threads to use during batch generation")
    parser.add_argument("--img_size", type=int, default=360, help="size of each image dimension")
    parser.add_argument("--checkpoint_interval", type=int, default=1, help="interval between saving model weights")
    parser.add_argument("--evaluation_interval", type=int, default=1, help="interval evaluations on validation set")
    parser.add_argument("--compute_map", default=False, help="if True computes mAP every tenth batch")
    parser.add_argument("--multiscale_training", default=True, help="allow for multi-scale training")
    opt = parser.parse_args()
```

训练模型

出现错误

更正：

```
# utils/logger.py
...
```

```
class Logger(object):
    def __init__(self, log_dir):
        self.writer = tf.summary.create_file_writer(log_dir)

    def scalar_summary(self, tag, value, step):
        """Log a scalar variable."""
        with self.writer.as_default():
            tf.summary.scalar(tag, value, step=step)

    def list_of_scalars_summary(self, tag_value_pairs, step):
        """Log scalar variables."""
        with self.writer.as_default():
            for tag, value in tag_value_pairs:
                tf.summary.scalar(tag, value, step=step)
```

出现错误

```
RuntimeError: MPS backend out of memory (MPS allocated: 116.47 GB, other allocations: 5.82 GB, max allowed: 122.40 GB). Tried to alloc
```

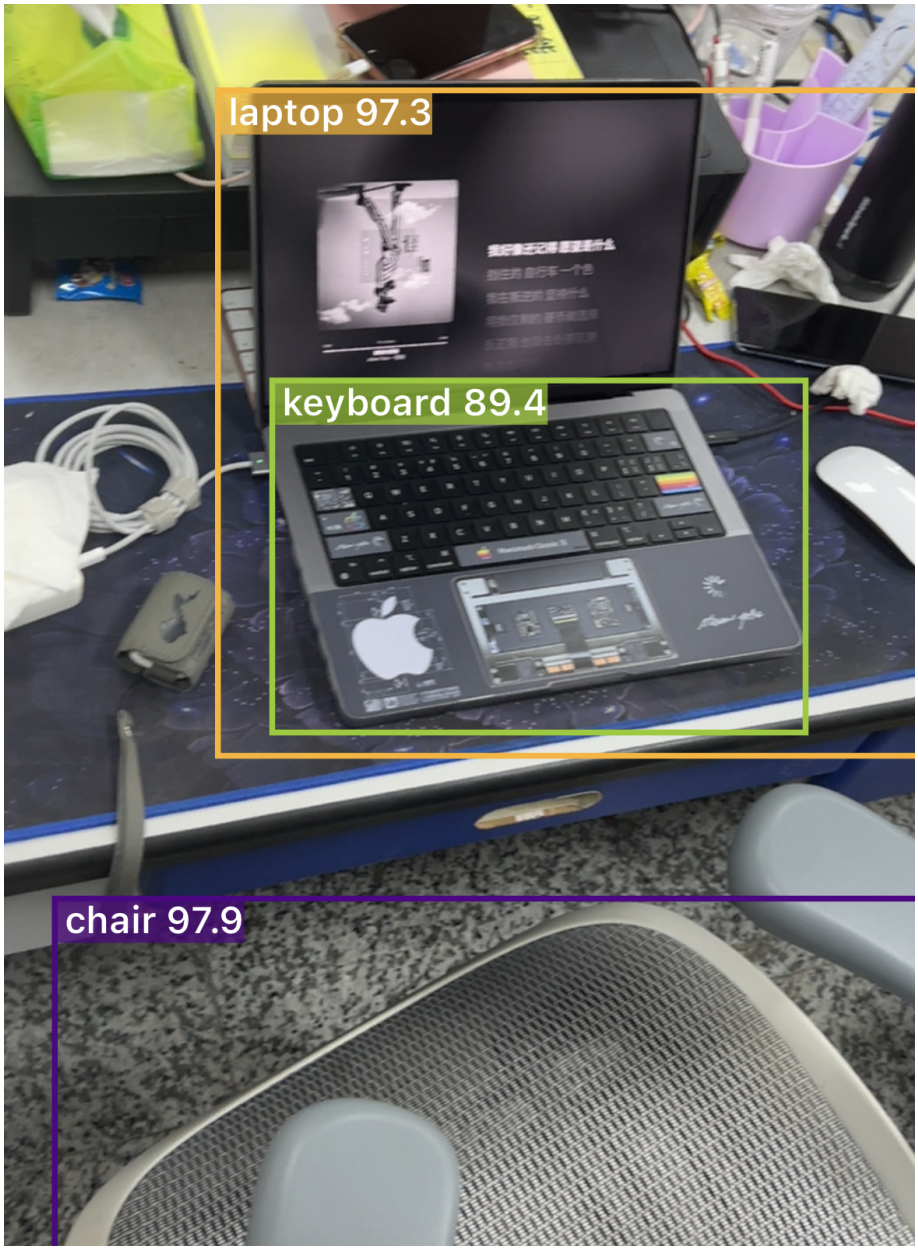
在环境中设置内存使用上限

```
export PYTORCH_MPS_HIGH_WATERMARK_RATIO=0.0
```

训练过程

```
---- [Epoch 0/3, Batch 0/587] ----
+------------+--------------+--------------+--------------+
| Metrics    | YOLO Layer 0 | YOLO Layer 1 | YOLO Layer 2 |
+------------+--------------+--------------+--------------+
| grid_size  | 11           | 22           | 44           |
| loss       | 84.450012    | 72.515976    | 71.529587    |
| x          | 0.094779     | 0.097502     | 0.092488     |
| y          | 0.099783     | 0.097919     | 0.092294     |
| w          | 2.925395     | 1.079291     | 0.713187     |
| h          | 1.439841     | 1.086689     | 1.513283     |
| conf       | 79.177940    | 69.434937    | 68.403503    |
| cls        | 0.712274     | 0.719635     | 0.714826     |
| cls_acc    | 0.60%        | 0.90%        | 0.84%        |
| recall50   | 0.000000     | 0.003003     | 0.001406     |
| recall75   | 0.000000     | 0.000000     | 0.000000     |
| precision  | 0.000000     | 0.000029     | 0.000004     |
| conf_obj   | 0.572391     | 0.488511     | 0.491140     |
| conf_noobj | 0.528962     | 0.491856     | 0.489770     |
+------------+--------------+--------------+--------------+
Total loss 228.49557495117188
---- ETA 1 day, 7:07:32.391346
```

结果展示


laptop 97.3
keyboard 89.4
chair 97.9