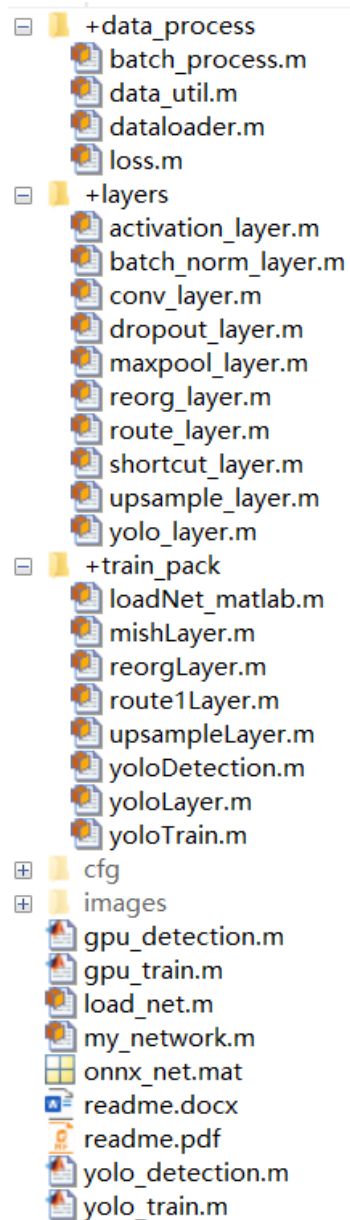


YOLO 的 Matlab 版代码说明

1 代码结构



1.1 自定义网络各层的前向推理

- (1) data_process 包：训练之前数据的预处理、损失函数的计算(一个 batch)等。
- (2) layer 包：神经网络的不同层，yolo、maxpool、conv 包括前向与反向，其他暂时只有前向。
- (3) cfg 目录：网络配置文件*.cfg，训练得到的权重文件*.weights，目标对象名称

文件等。

(4) images: 检测的图像。

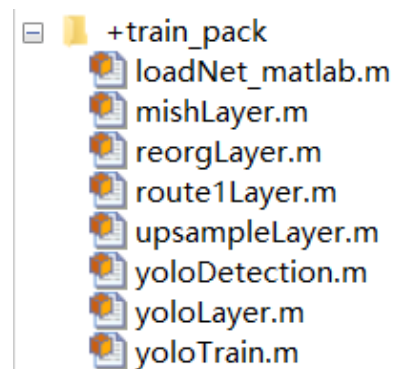
(5) load_net.m: 通过 cfg 加载网络。

(6) my_network.m: 串联所有层的网络。

(7) yolo_detection.m: 目标检测。

(8) yolo_train.m: 网络训练。

1.2 依赖 Matlab 深度学习算子的检测与训练



(1) train_pack 包: 包括神经网络的各层, 加载网络与权重(适配 Matlab 深度学习算子), 检测与训练处理等。

loadNet_matlab.m: 通过 cfg 文件、weights 文件加载网络(适配 Matlab 算子)。包括两种方式: 1. fun(cfg_file, weight_file), 适用于训练与检测。训练时 weight_file 为主干网络预训练权重, 检测时 weight_file 为某个网络训练好的权重; 2. fun(cfg_file)只适用训练时, 没有加载预训练权重, 通过自定义权重初始化。

yoloDetection.m: 通过 gpu 检测一个目录下的所有图像。其中包括非极大值抑制等。

yoloTrain.m: 通过 gpu 做训练。其中包括损失函数计算, 反向传播及梯度更新等。

(2) gpu_detection: 使用 GPU 做检测的入口。

(3) gpu_train: 使用 GPU 做训练的入口。

2 检测部分

包括：yolov2、yolov3、yolo-fastest、yolo-lite。

```
yolo_detection.m x +
tic

% img_path = './images/zidane.jpg';
% img_path = './images/dog.jpg';
img_path = './images/field.jpg';
% img_path = './images/horses.jpg';

%===== yolov2 =====%
% mn = my_network(img_path,'onnx_net.mat');
% mn = my_network(img_path,'cfg/yolov2/yolov2.cfg','cfg/yolov2/yolov2.weights');
% mn = my_network(img_path,'cfg/yolov2/yolov2-tiny.cfg','cfg/yolov2/yolov2-tiny.weights');

%===== yolov3 =====%
% mn = my_network(img_path,'cfg/yolov3/yolov3.cfg','cfg/yolov3/yolov3.weights');
% mn = my_network(img_path,'cfg/yolov3/yolov3-tiny.cfg','cfg/yolov3/yolov3-tiny.weights');

%===== yolo-fastest =====%
% mn = my_network(img_path,'cfg/fastest/yolo-fastest.cfg','cfg/fastest/yolo-fastest.weights');
% mn = my_network(img_path,'cfg/fastest/yolo-fastest-xl.cfg','cfg/fastest/yolo-fastest-xl.weights');
% mn = my_network(img_path,'cfg/fastest/yolo-fastest-1.1.cfg','cfg/fastest/yolo-fastest-1.1.weights');
mn = my_network(img_path,'cfg/fastest/yolo-fastest-1.1-xl.cfg','cfg/fastest/yolo-fastest-1.1-xl.weights');

%===== yolov2-lite =====%
% mn = my_network(img_path,'cfg/lite/tiny-yolov2-trial3-noBatch.cfg','cfg/lite/tiny-yolov2-trial3-noBatch.weights');
% mn = my_network(img_path,'cfg/lite/trial6.cfg','cfg/lite/trial6_653550.weights');
```

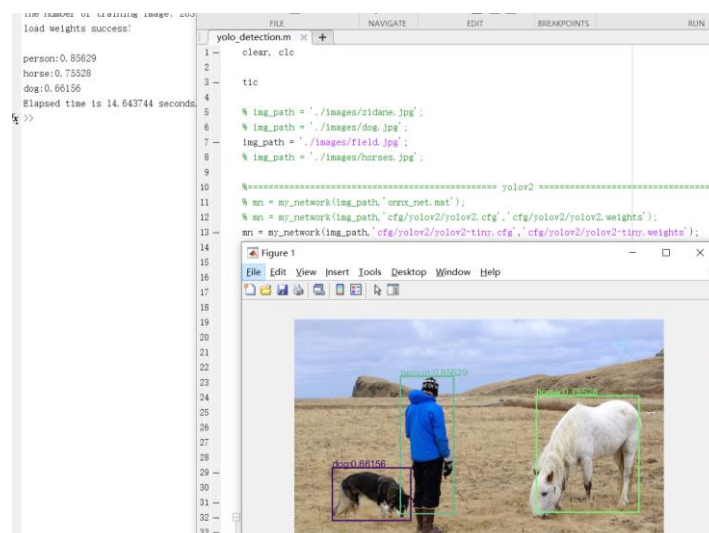
第一部分加载图像，第二部分加载网络、权重、图像。

2.1 yolov2

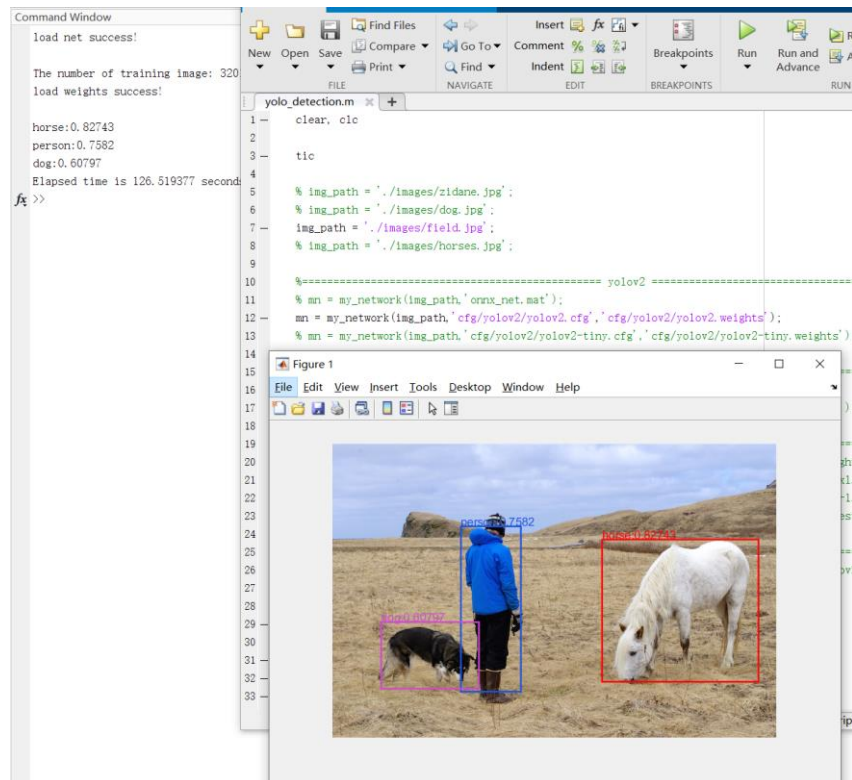
加载网络存在两种方式：mat 文件、cfg+weights

```
% mn = my_network(img_path,'onnx_net.mat');
% mn = my_network(img_path,'cfg/yolov2/yolov2.cfg','cfg/yolov2/yolov2.weights');
% mn = my_network(img_path,'cfg/yolov2/yolov2-tiny.cfg','cfg/yolov2/yolov2-tiny.weights');
```

2.1.1 yolov2-tiny

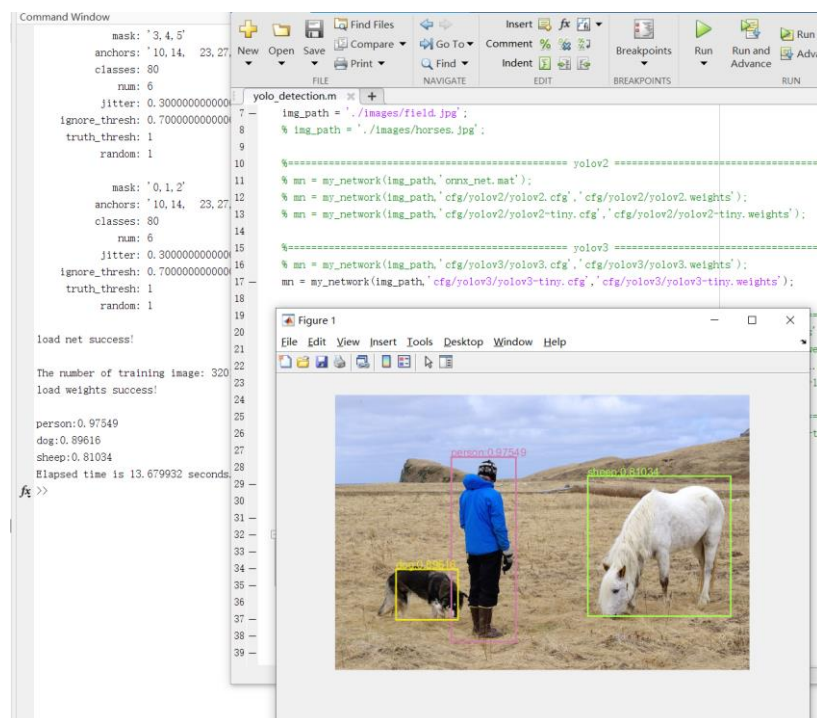


2.1.2 yolov2



2.2 yolov3

2.2.1 yolov3-tiny



2.2.2 yolov3

```
mask: '6, 7, 8'
anchors: '10, 13, 16, 30'
classes: 80
num: 9
jitter: 0.30000000000000004
ignore_thresh: 0.7000000000000001
truth_thresh: 1
random: 1

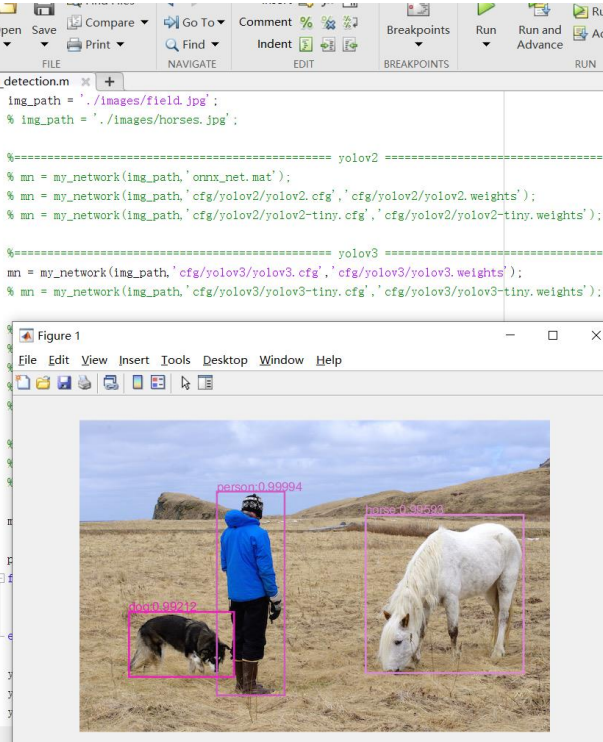
mask: '3, 4, 5'
anchors: '10, 13, 16, 30'
classes: 80
num: 9
jitter: 0.30000000000000004
ignore_thresh: 0.7000000000000001
truth_thresh: 1
random: 1

load net success!

The number of training image: 320
load weights success!

person:0.99994
horse:0.99593
dog:0.99212
Elapsed time is 98.933446 seconds.
fx >>
```

```
7 img_path = './images/field.jpg';
8 % img_path = './images/horses.jpg';
9
10
11 %===== yolov2 =====
12 mn = my_network(img_path,'ormx_net.mat');
13 mn = my_network(img_path,'cfg/yolov2/yolov2.cfg','cfg/yolov2/yolov2.weights');
14 mn = my_network(img_path,'cfg/yolov2/yolov2-tiny.cfg','cfg/yolov2/yolov2-tiny.weights');
15
16 %===== yolov3 =====
17 mn = my_network(img_path,'cfg/yolov3/yolov3.cfg','cfg/yolov3/yolov3.weights');
18 mn = my_network(img_path,'cfg/yolov3/yolov3-tiny.cfg','cfg/yolov3/yolov3-tiny.weights');
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```



2.3 yolo-fastest

```
mask: '3, 4, 5'
anchors: '12, 18, 37'
classes: 80
num: 6
jitter: 0.15000000000000002
truth_thresh: 1
random: 0
scale_x_y: 1
iou_thresh: 0.21300000000000003
cls_normalizer: 1
iou_normalizer: 0.07000000000000001
iou_loss: 'ciou'
nms_kind: 'greedy_nms'
beta_nms: 0.6000000000000001

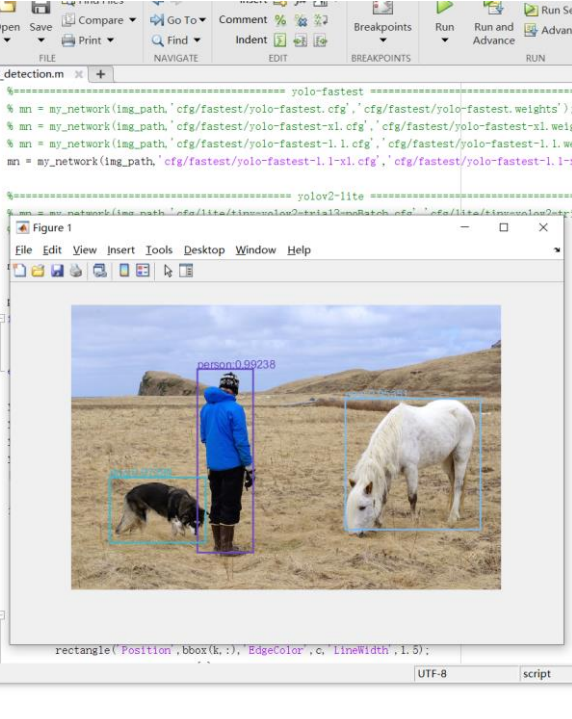
mask: '0, 1, 2'
anchors: '12, 18, 37'
classes: 80
num: 6
jitter: 0.15000000000000002
ignore_thresh: 0.5000000000000001
truth_thresh: 1
random: 0
scale_x_y: 1
iou_thresh: 0.21300000000000003
cls_normalizer: 1
iou_normalizer: 0.07000000000000001
iou_loss: 'ciou'
nms_kind: 'greedy_nms'
beta_nms: 0.6000000000000001

load net success!

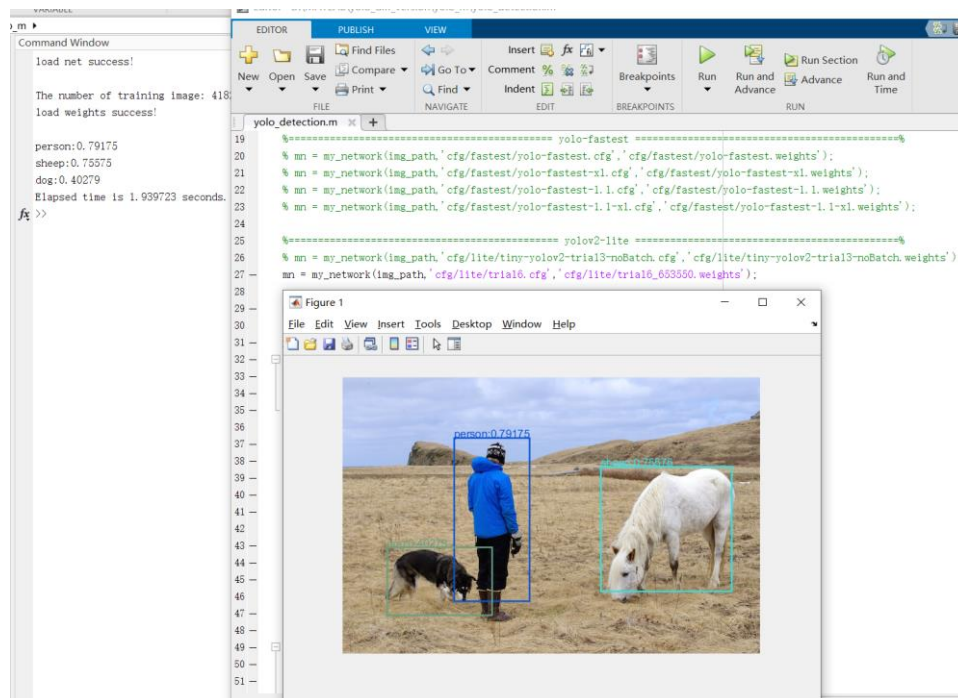
The number of training image: 160
load weights success!

person:0.99238
dog:0.97299
cow:0.95351
Elapsed time is 5.023959 seconds.
fx >>
```

```
20 mn = my_network(img_path,'cfg/fastest/yolo-fastest.cfg','cfg/fastest/yolo-fastest.weights');
21 mn = my_network(img_path,'cfg/fastest/yolo-fastest-xl.cfg','cfg/fastest/yolo-fastest-xl.weights');
22 mn = my_network(img_path,'cfg/fastest/yolo-fastest-l.1.cfg','cfg/fastest/yolo-fastest-l.1.weights');
23 mn = my_network(img_path,'cfg/fastest/yolo-fastest-l.1-xl.cfg','cfg/fastest/yolo-fastest-l.1-xl.weights');
24
25 %===== yolov2-tiny =====
26 mn = my_network(img_path,'cfg/yolov2/yolov2-tiny.cfg','cfg/yolov2/yolov2-tiny.weights');
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```

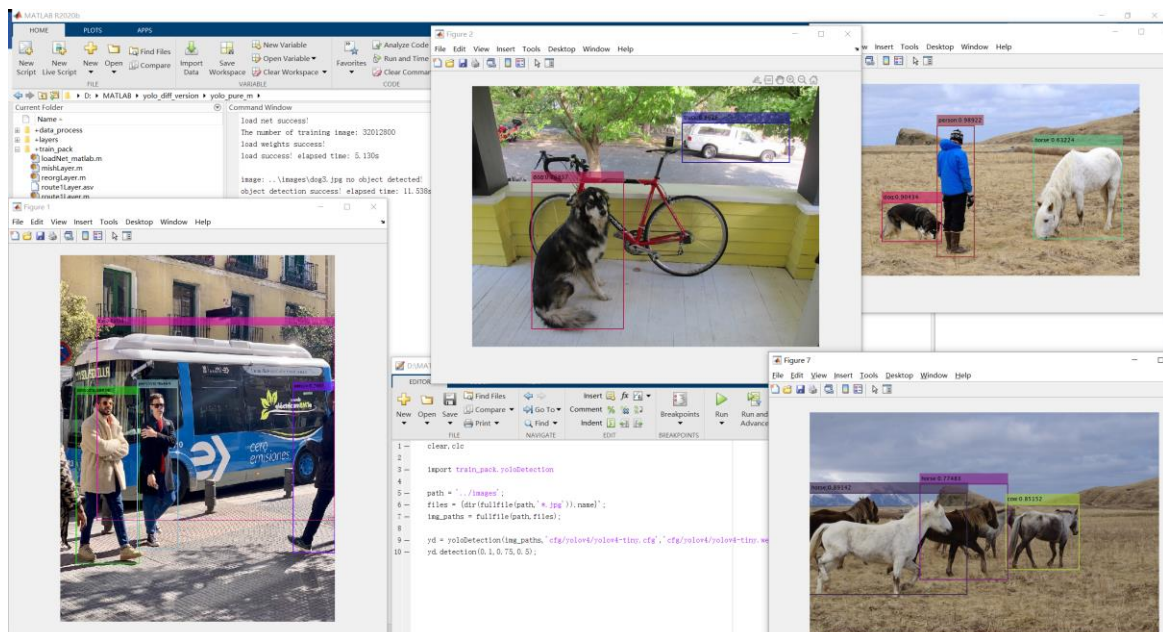


2.4 yolo-lite

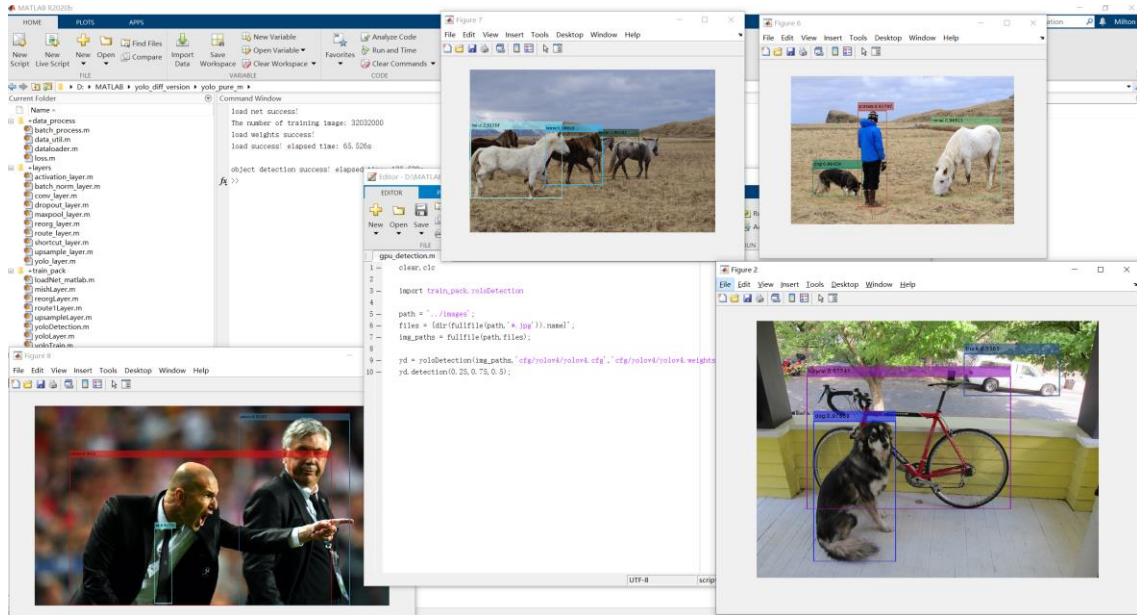


2.5 yolov4(GPU)

2.5.1 yolov4-tiny



2.5.2 yolov4



3 训练部分

训练部分暂时采用 lite 做训练,其中没有包含 batch_norm。数据集采用 coco, coco 数据集的配置如下:

```
FILE NAVIGATE
yolo_train.m x coco.data x +
classes= 80
train=E:/DataSets/COCO/trainvalno5k_.txt
valid=E:/DataSets/COCO/5k_.txt
names=data/coco.names
backup=backup/
eval=coco
```

另外,反向传播目前只存在卷积层(conv_layer)、最大池化层(maxpool_layer)、yolo 层(yolo_layer)。训练过程中添加了并行运算,主要是针对 for 循环。

3.1 Lite 网络训练

一个 batch 中包括 8 张图像。计算了 IoU、Object、Class 与 Total、Batch 损失。训练完成之后把网络保存为 mat 文件。

```
load net success!

Batch 1
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 8).
YOL0 1: [IoU loss: 1.991, Object loss: 21.015, Class loss: 3.984, Loss: 26.990, Batch loss: 215.921879]
Elapsed time: 20.533s.
Backward elapsed time: 59.980s.
Batch 2
YOL0 1: [IoU loss: 2.080, Object loss: 16.192, Class loss: 3.261, Loss: 21.534, Batch loss: 172.270165]
Elapsed time: 3.872s.
Backward elapsed time: 61.336s.
Batch 3
YOL0 1: [IoU loss: 2.413, Object loss: 8.656, Class loss: 3.748, Loss: 14.818, Batch loss: 118.543740]
Elapsed time: 3.674s.
Backward elapsed time: 58.196s.
Batch 4
YOL0 1: [IoU loss: 3.277, Object loss: 12.100, Class loss: 3.098, Loss: 18.475, Batch loss: 147.798593]
Elapsed time: 3.396s.
```


3.2 基于 Matlab 深度学习算子的训练

通过运行 `gpu_train.m` 来训练网络，`yoloTrain` 存在两种调用方式：
`yoloTrain(cfg_file, weight_file, data_file)`，或者 `yoloTrain(cfg_file, data_file)`。即是否加载预训练权重。

```
gpu_train.m x +
1 — clear, clc
2
3 — import train_pack.yoloTrain
4
5 — yt = yoloTrain('cfg/yolov3/yolov3.cfg', 'cfg/coco.data');
6 — yt.train;
7
```

训练效果：

```
load net success!
load net and init parm elapsed time: 9.428s.
The loss result[1][8/117264]: [IoU loss: 2.131, Object loss: 894.154, Class loss: 1.351, Loss: 897.637, Batch loss: 7181.097373]
Time left: 232 day, 11 hours, 8 minutes, 9.509 seconds.
The loss result[1][16/117264]: [IoU loss: 2.420, Object loss: 46.732, Class loss: 0.911, Loss: 50.063, Batch loss: 400.503763]
Time left: 287 day, 2 hours, 38 minutes, 53.796 seconds.
The loss result[1][24/117264]: [IoU loss: 1.867, Object loss: 5.682, Class loss: 1.199, Loss: 8.748, Batch loss: 69.983251]
Time left: 337 day, 19 hours, 46 minutes, 4.884 seconds.
The loss result[1][32/117264]: [IoU loss: 2.520, Object loss: 6.855, Class loss: 0.887, Loss: 10.262, Batch loss: 82.099597]
Time left: 319 day, 6 hours, 58 minutes, 20.141 seconds.
The loss result[1][40/117264]: [IoU loss: 1.947, Object loss: 3.730, Class loss: 0.688, Loss: 6.365, Batch loss: 50.920824]
Time left: 328 day, 3 hours, 51 minutes, 43.970 seconds.
The loss result[1][48/117264]: [IoU loss: 2.244, Object loss: 4.637, Class loss: 1.036, Loss: 7.916, Batch loss: 63.330568]
Time left: 320 day, 8 hours, 24 minutes, 52.347 seconds.
The loss result[1][56/117264]: [IoU loss: 2.024, Object loss: 4.138, Class loss: 0.968, Loss: 7.130, Batch loss: 57.040628]
Time left: 321 day, 17 hours, 34 minutes, 15.045 seconds.
The loss result[1][64/117264]: [IoU loss: 1.972, Object loss: 3.075, Class loss: 1.263, Loss: 6.310, Batch loss: 50.476523]
Time left: 320 day, 14 hours, 37 minutes, 38.526 seconds.
```

说明：[epoch][img_num/img_total_num]，三种损失与一个 batch 的总损失。

训练剩余的大概时间

