Yolov3

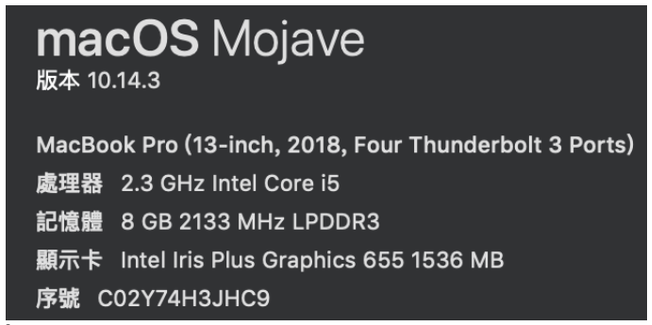
Vincent Wu

- **Topic:**
  Recognition kangaroo and raccoon

- **Abstract:**
  Use the tensorflow yolov3 by YunYang to recognize the kangaroo and raccoon. Encountered some difficulty about syntax change on different tensorflow version, poor prediction results by low confidence value but low loss value and some bug in this reference Tensorflow yolov3. Some is fixed, but low confidence value is improved a little. Even the model performance is not good to recognize the picture, but I got a lot in this final project.

- **Code reference** (tensorflow yolov3, not Keras yolov3)
  Reference Github:
  https://github.com/YunYang1994/tensorflow-yolov3
  My project GitHub:
  https://github.com/double1010x2/1st-DL-CVMarathon/tree/master/homework/finalProject/tensorflow-yolov3

- **PC resource:**



- **Difficulty:**
  (1) Debugging the code error
  (2) Low confidence value when model prediction
  (3) syntax different between different tensorflow version

- **Code function:**
  kmeans_anchor.py → clustering the anchor information for kangaroo and raccoon data
  xml_to_csv.py. → collect xml file to csv and txt file (anchor format: *.jpg xmin,xmax,ymin,ymax,class
  train.py → training the yolov3 with darknet53 network (in core/yolov3.py)
  freeze_graph.py. → transfer *ckpt file to *.pb file #python freeze_graph.py *.ckpt
  image_demo.py → prediction yolov3 model on one image #python image_demo.py *.pb *.jpg
  video_demo.py → prediction yolov3 model on one video #python image_demo.py *.pb *.mp4

Training data: (raccoon+kangaroo) 80%
Test data: (raccoon+kangaroo) 20%

Data preparation

Download image/annotations

Raccoon data :
images: *https://github.com/experiencor/raccoon_dataset/tree/master/images*
Annotation: *https://github.com/experiencor/raccoon_dataset/tree/master/annotations*

Kangaroo data :
images: *https://github.com/experiencor/kangaroo/tree/master/images*
Annotation: *https://github.com/experiencor/kangaroo/tree/master/annots*

Collect image/ annotations to txt

Xml to dataframe to csv and txt

```
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text),
                     1 if member[0].text == "kangaroo" else 0
                     )
            xml_list.append(value)
    #column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    column_name = ['filename', 'xmin', 'xmax', 'ymin', 'ymax', 'class']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    image_path  = path.replace('annotations', 'images')
    xml_df.filename = image_path + xml_df.filename.values
    return xml_df
```

Txt format: *.jpg xmin,ymin,xman,ymax,class

```
00175.jpg 230,66,534,449,1
00013.jpg 223,87,428,355,1
```

Check the best anchor ratio on this data by kmeans

```
data, data_size = load_dataset(ANNOTATIONS_PATH)
out = kmeans(data, k=CLUSTERS)
```

(kmens_anchor.py)

```
print("Accuracy: {:.2f}%".format(avg_iou(data, out) * 100))
```

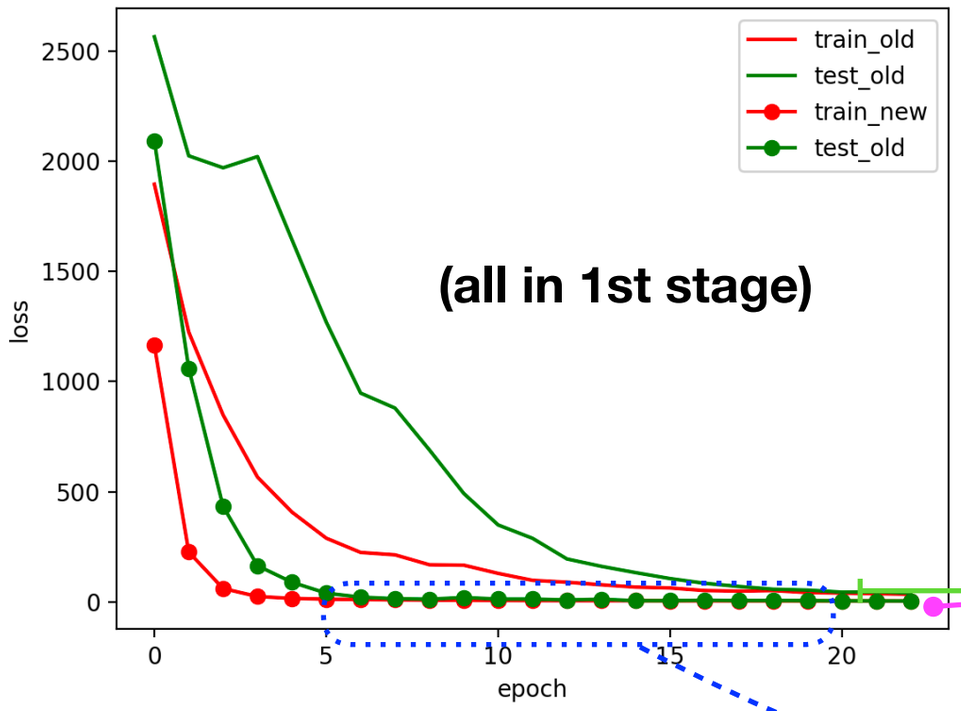
```
Accuracy: 79.12%
```

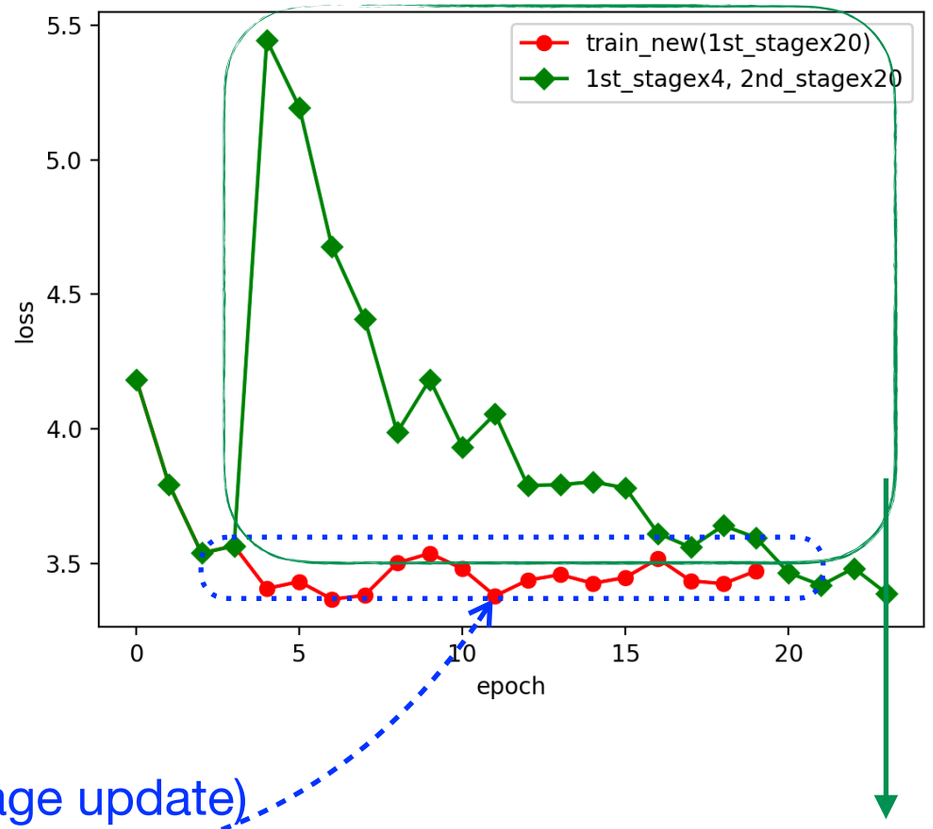
https://blog.csdn.net/cgt19910923/article/details/82154401

Hyper-parameter setting (core/config.py)

```
# Set the class name
__C.YOLO.CLASSES                = "/Users/vincentwu/Documents/GitHub/1st-DL-CVMarath
__C.YOLO.ANCHORS                = "./data/anchors/basline_anchors.txt"
#__C.YOLO.ANCHORS                 = "./data/anchors/coco_anchors.txt"
__C.YOLO.MOVING_AVE_DECAY       = 0.9995
__C.YOLO.STRIDES                = [8, 16, 32]
__C.YOLO.ANCHOR_PER_SCALE       = 3
__C.YOLO.IOU_LOSS_THRESH        = 0.5
__C.YOLO.UPSAMPLE_METHOD        = "resize"
__C.YOLO.ORIGINAL_WEIGHT        = "./checkpoint/yolov3_coco.ckpt"
__C.YOLO.DEMO_WEIGHT            = "./checkpoint/yolov3_coco_demo.ckpt"

# Train options
__C.TRAIN                       = edict()

#__C.TRAIN.ANNOT_PATH             = "./data/dataset/voc_train.txt"
__C.TRAIN.ANNOT_PATH            = "/Users/vincentwu/Documents/GitHub/1st-DL-CVMarath
__C.TRAIN.BATCH_SIZE            = 8    (old = 16)
__C.TRAIN.INPUT_SIZE            = [320, 352, 384, 416, 448, 480, 512, 544, 576, 608]
#__C.TRAIN.INPUT_SIZE             = [352, 352, 384, 416, 448]
__C.TRAIN.DATA_AUG              = True
__C.TRAIN.LEARN_RATE_INIT       = 5e-4 (old = 1e-4)
__C.TRAIN.LEARN_RATE_END        = 1e-6
__C.TRAIN.WARMUP_EPOCHS         = 2
__C.TRAIN.FISRT_STAGE_EPOCHS    = 20
__C.TRAIN.SECOND_STAGE_EPOCHS   = 40
#__C.TRAIN.INITIAL_WEIGHT         = "./checkpoint/yolov3_coco_demo.ckpt"
__C.TRAIN.INITIAL_WEIGHT        = "./checkpoint/yolov3_test_loss=26.7363.pb-29"
```

hyper-parameter change results
→convergence faster





(add 2nd stage update)

Learning rate: 1e-4→5e-4
Mini - batch: 16. →8
Anchors: coco_anchors→ baseline_anchors (by kmeans)

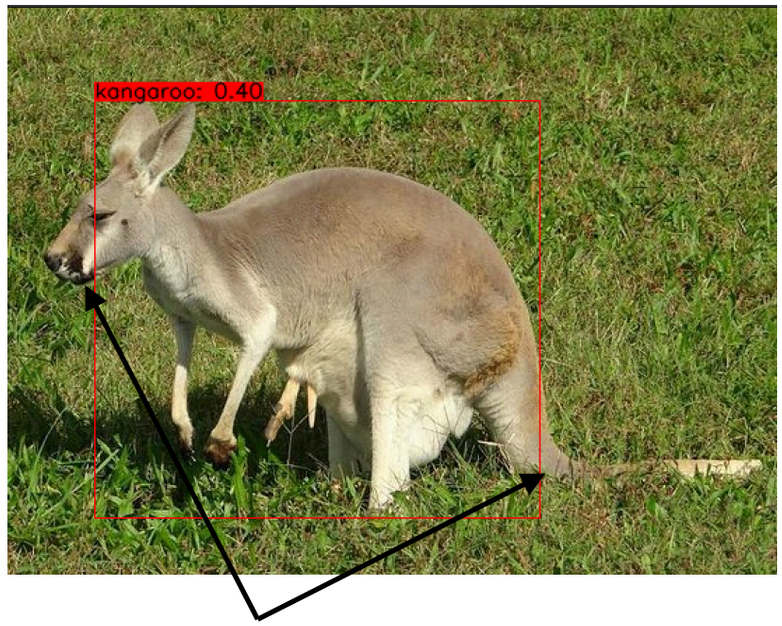
Augment your batch: better training with larger batches

Elad Hoffer [1 2] Tal Ben-Nun [3] Itay Hubara [1 2] Niv Giladi [1] Torsten Hoefler [3] Daniel Soudry [1]

Owing to the study, we larger the batch size from 8 to 32 for better convergence and stable Unfortunately , the model should go head to improve it's convergence, small epoch count

Recognition results - Kangaroo

- Kangaroo is alone



A little weird. Right one should be better recognized results due to large RGB different difference between animal and environment.

A little Cropped

- Kangaroo is not alone

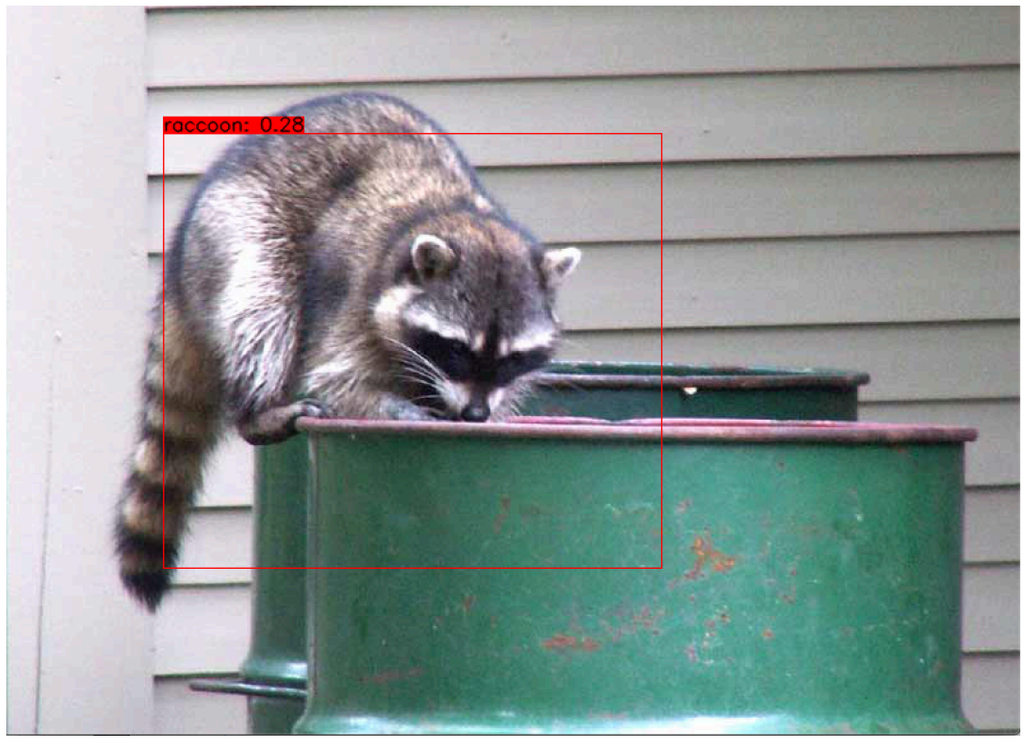
I need to low the IOU value due to confidence value in order to show that picture.

A group of kangaroo is not good, should change IOU value?

Recognition results - Raccoon

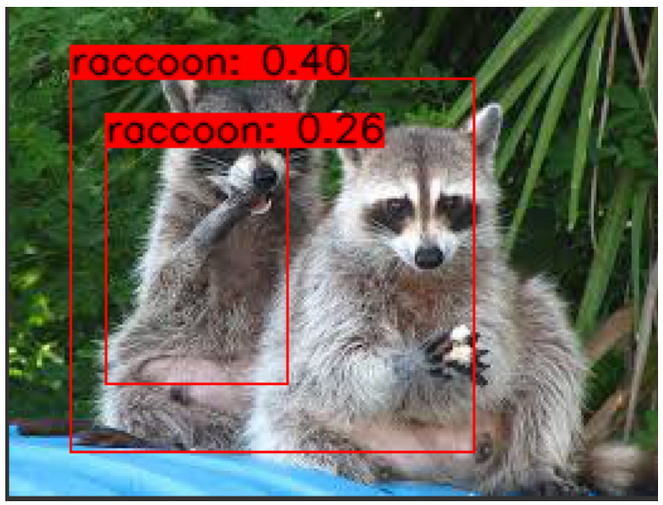
- Raccoon is alone

Picture with Small size is fine



Picture with large size is a little cropped and shift, and confident value is low

- Raccoon is not alone



I need to low the IOU value due to confidence value in order to show that picture.

Two animals would get poor recognition. In my opinion, model stopped iteration at epoch 20th and should go on to improve model's classify ability~

Recognition results - video

Environment clean would got a marker but low confidence value

Due to low IOU value, the maker would be false alarm

The results as the same as previous single image

Training code struct

config.py

Global parameter setting (in core folder)

*: introduce later

Train.py*

core folder

yolov3.py* Define yolov3 network

utils.py Utility function

dataset.py* Data Iterator, data augmentation

common.py Tensorflow utility function for network

backbone.py Define darknet53 network

Checkpoint folder

Save model file *.ckpt

class
(class list)

anchor
(anchor ratio by kmeans)

dataset
(annotation/image)

Data folder

train.py

Train.py

```python
with tf.name_scope("define_first_stage_train"):
    self.first_stage_trainable_var_list = []
    for var in tf.trainable_variables():
        var_name = var.op.name
        var_name_mess = str(var_name).split('/')
        if var_name_mess[0] in ['conv_sbbox', 'conv_mbbox', 'conv_lbbox
            self.first_stage_trainable_var_list.append(var)
```

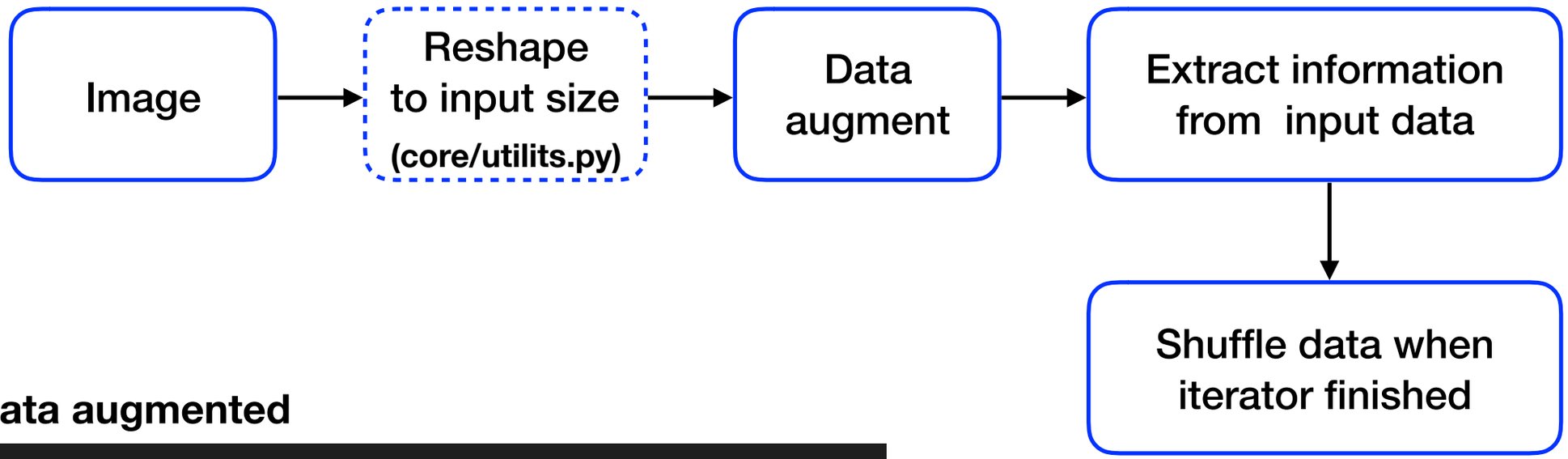
Only update three box parameter

```python
def train(self):
    self.sess.run(tf.global_variables_initializer())
    try:
        print('=> Restoring weights from: %s ... ' % self.initial_weight)
        self.loader.restore(self.sess, self.initial_weight)
    except:
        print('=> %s does not exist !!!' % self.initial_weight)
        print('=> Now it starts to train YOLOV3 from scratch ...')
        self.first_stage_epochs = 0

    for epoch in range(1, 1+self.first_stage_epochs+self.second_stage_epochs):
        if epoch <= self.first_stage_epochs:
            train_op = self.train_op_with_frozen_variables
        else:
            train_op = self.train_op_with_all_variables
        #    self.trainset.batch_size *= 4
        #    self.testset.batch_size *= 4

        pbar = tqdm(self.trainset)
        train_epoch_loss, test_epoch_loss = [], []
        train_loss, test_loss = [], []

        for train_data in pbar:
            _, summary, train_step_loss, global_step_val = self.sess.run(
                [train_op, self.write_op, self.loss, self.global_step],feed_dict={
                                            self.input_data:   train_data[0],
                                            self.label_sbbox:  train_data[1],
                                            self.label_mbbox:  train_data[2],
                                            self.label_lbbox:  train_data[3],
                                            self.true_sbboxes: train_data[4],
                                            self.true_mbboxes: train_data[5],
                                            self.true_lbboxes: train_data[6],
                                            self.trainable:    True,
```

Load retrain model by *.ckpt file

One of root cause about Low confidential value !!!

core/dataset.py



Data augmented

```
if self.data_aug:
    image, bboxes = self.random_horizontal_flip(np.copy(image), np.copy(bboxes))
    image, bboxes = self.random_crop(np.copy(image), np.copy(bboxes))
    image, bboxes = self.random_translate(np.copy(image), np.copy(bboxes))
```

Extract information from input data

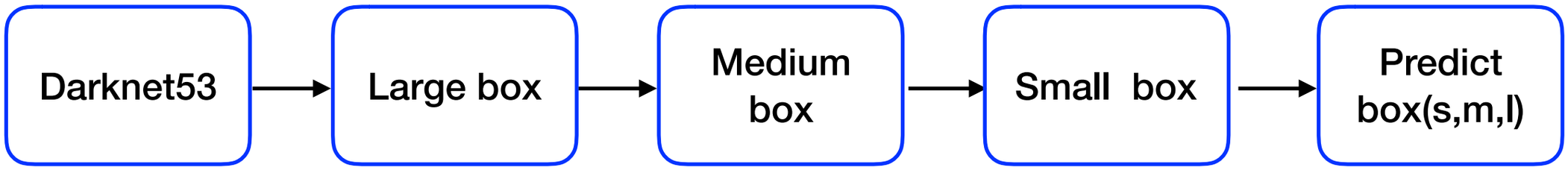
```
annotation = self.annotations[index]
image, bboxes = self.parse_annotation(annotation)
label_sbbox, label_mbbox, label_lbbox, sbboxes, mbboxes, lbboxes = self.preprocess_true_boxes(bboxes)
```

```
def preprocess_true_boxes(self, bboxes):

    label = [np.zeros((self.train_output_sizes[i], self.train_output_sizes[i], self.anchor_per_scale,
                       5 + self.num_classes)) for i in range(3)]
    bboxes_xywh = [np.zeros((self.max_bbox_per_scale, 4)) for _ in range(3)]
    bbox_count = np.zeros((3,))

    for bbox in bboxes:
        bbox_coor = bbox[:4]
        bbox_class_ind = bbox[4]

        onehot = np.zeros(self.num_classes, dtype=np.float)
        onehot[bbox_class_ind] = 1.0
        uniform_distribution = np.full(self.num_classes, 1.0 / self.num_classes)
        deta = 0.01
        smooth_onehot = onehot * (1 - deta) + deta * uniform_distribution

        bbox_xywh = np.concatenate([(bbox_coor[2:] + bbox_coor[:2]) * 0.5, bbox_coor[2:] - bbox_coor[:2]], axis=-1)
        bbox_xywh_scaled = 1.0 * bbox_xywh[np.newaxis, :] / self.strides[:, np.newaxis]

        iou = []
        exist_positive = False
        for i in range(3):
            anchors_xywh = np.zeros((self.anchor_per_scale, 4))
            anchors_xywh[:, 0:2] = np.floor(bbox_xywh_scaled[i, 0:2]).astype(np.int32) + 0.5
            anchors_xywh[:, 2:4] = self.anchors[i]

            iou_scale = self.bbox_iou(bbox_xywh_scaled[i][np.newaxis, :], anchors_xywh)
            iou.append(iou_scale)
            iou_mask = iou_scale > float(cfg.YOLO.IOU_LOSS_THRESH)
```

Data shuffle when 1 epoch finished

```
else:
    self.batch_count = 0
    np.random.shuffle(self.annotations)
    raise StopIteration
```

core/yolov3.py(network + loss define)




```
def __build_nework(self, input_data):

    route_1, route_2, input_data = backbone.darknet53(input_data, self.trainable)
```

Large box

```
input_data = common.convolutional(input_data, (1, 1, 1024,  512), self.trainable, 'conv52')
input_data = common.convolutional(input_data, (3, 3,  512, 1024), self.trainable, 'conv53')
input_data = common.convolutional(input_data, (1, 1, 1024,  512), self.trainable, 'conv54')
input_data = common.convolutional(input_data, (3, 3,  512, 1024), self.trainable, 'conv55')
input_data = common.convolutional(input_data, (1, 1, 1024,  512), self.trainable, 'conv56')

conv_lobj_branch = common.convolutional(input_data, (3, 3, 512, 1024), self.trainable, name='conv_lobj_branch')
conv_lbbox = common.convolutional(conv_lobj_branch, (1, 1, 1024, 3*(self.num_class + 5)),
                                  trainable=self.trainable, name='conv_lbbox', activate=False, bn=False)
```

Medium box

Only update when epoch smaller than `__C.TRAIN.FISRT_STAGE_EPOCHS = 20`

```
input_data = common.convolutional(input_data, (1, 1, 768, 256), self.trainable, 'conv58')
input_data = common.convolutional(input_data, (3, 3, 256, 512), self.trainable, 'conv59')
input_data = common.convolutional(input_data, (1, 1, 512, 256), self.trainable, 'conv60')
input_data = common.convolutional(input_data, (3, 3, 256, 512), self.trainable, 'conv61')
input_data = common.convolutional(input_data, (1, 1, 512, 256), self.trainable, 'conv62')

conv_mobj_branch = common.convolutional(input_data, (3, 3, 256, 512),  self.trainable, name='conv_mobj_branch' )
conv_mbbox = common.convolutional(conv_mobj_branch, (1, 1, 512, 3*(self.num_class + 5)),
                                  trainable=self.trainable, name='conv_mbbox', activate=False, bn=False)
```

Small box

```
input_data = common.convolutional(input_data, (1, 1, 384, 128), self.trainable, 'conv64')
input_data = common.convolutional(input_data, (3, 3, 128, 256), self.trainable, 'conv65')
input_data = common.convolutional(input_data, (1, 1, 256, 128), self.trainable, 'conv66')
input_data = common.convolutional(input_data, (3, 3, 128, 256), self.trainable, 'conv67')
input_data = common.convolutional(input_data, (1, 1, 256, 128), self.trainable, 'conv68')

conv_sobj_branch = common.convolutional(input_data, (3, 3, 128, 256), self.trainable, name='conv_sobj_branch')
conv_sbbox = common.convolutional(conv_sobj_branch, (1, 1, 256, 3*(self.num_class + 5)),
                                  trainable=self.trainable, name='conv_sbbox', activate=False, bn=False)
```

Decode: let box after network to be predict box

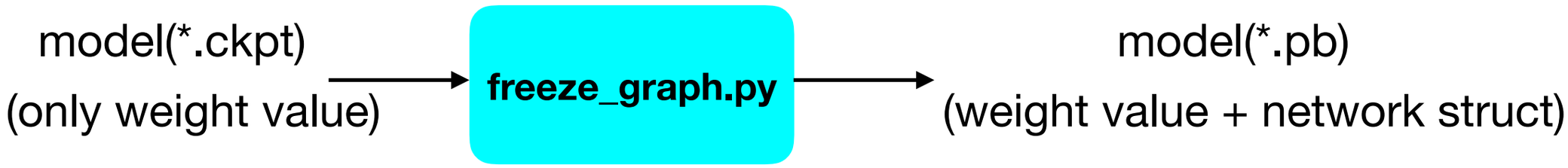
```
with tf.variable_scope('pred_sbbox'):
    self.pred_sbbox = self.decode(self.conv_sbbox, self.anchors[0], self.strides[0])
```

Loss define in self.loss_layers

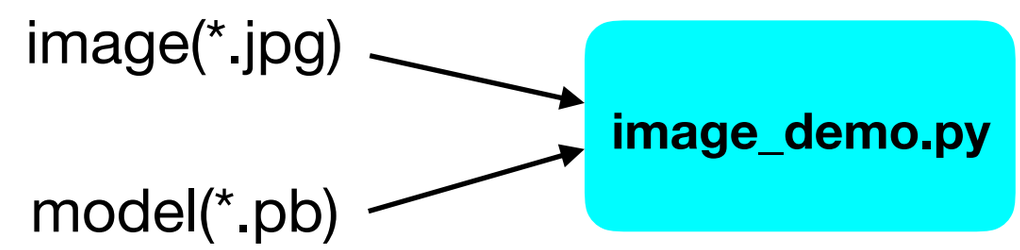
```
giou = tf.expand_dims(self.bbox_giou(pred_xywh, label_xywh), axis=-1)
input_size = tf.cast(input_size, tf.float32)

bbox_loss_scale = 2.0 - 1.0 * label_xywh[:, :, :, :, 2:3] * label_xywh[:, :, :, :, 3:4] / (input_size ** 2)
giou_loss = respond_bbox * bbox_loss_scale * (1- giou)

iou = self.bbox_iou(pred_xywh[:, :, :, :, np.newaxis, :], bboxes[:, np.newaxis, np.newaxis, np.newaxis, :, :])
max_iou = tf.expand_dims(tf.reduce_max(iou, axis=-1), axis=-1)

respond_bgd = (1.0 - respond_bbox) * tf.cast( max_iou < self.iou_loss_thresh, tf.float32 )

conf_focal = self.focal(respond_bbox, pred_conf)

conf_loss = conf_focal * (
        respond_bbox * tf.nn.sigmoid_cross_entropy_with_logits(labels=respond_bbox, logits=conv_raw_conf)
        +
        respond_bgd * tf.nn.sigmoid_cross_entropy_with_logits(labels=respond_bbox, logits=conv_raw_conf)
)

prob_loss = respond_bbox * tf.nn.sigmoid_cross_entropy_with_logits(labels=label_prob, logits=conv_raw_prob)

giou_loss = tf.reduce_mean(tf.reduce_sum(giou_loss, axis=[1,2,3,4]))
conf_loss = tf.reduce_mean(tf.reduce_sum(conf_loss, axis=[1,2,3,4]))
prob_loss = tf.reduce_mean(tf.reduce_sum(prob_loss, axis=[1,2,3,4]))
```

Verify model

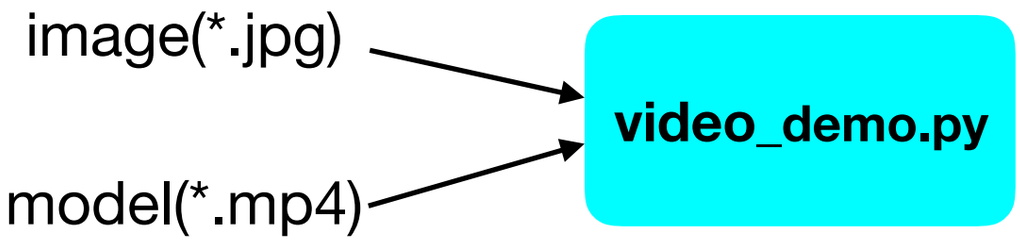
Transfer model *.ckpt to *.pb (



Recognize one image by model



Recognize one video by model



Difficulty

- (1) frozen IOU value, 0.3, in hard code (core/dataset.py)

```
215                 for i in range(3):
216                     anchors_xywh = np.zeros((self.anchor_per_scale, 4))
217                     anchors_xywh[:, 0:2] = np.floor(bbox_xywh_scaled[i, 0:2]).astype(np.int32) + 0.5
218                     anchors_xywh[:, 2:4] = self.anchors[i]
219
220                     iou_scale = self.bbox_iou(bbox_xywh_scaled[i][np.newaxis, :], anchors_xywh)
221                     iou.append(iou_scale)
222                     iou_mask = iou_scale > 0.3
```

```
iou = []
exist_positive = False
for i in range(3):
    anchors_xywh = np.zeros((self.anchor_per_scale, 4))
    anchors_xywh[:, 0:2] = np.floor(bbox_xywh_scaled[i, 0:2]).astype(np.int32) + 0.5
    anchors_xywh[:, 2:4] = self.anchors[i]

    iou_scale = self.bbox_iou(bbox_xywh_scaled[i][np.newaxis, :], anchors_xywh)
    iou.append(iou_scale)
    iou_mask = iou_scale > float(cfg.YOLO.IOU_LOSS_THRESH)

    if np.any(iou_mask):
        xind, yind = np.floor(bbox_xywh_scaled[i, 0:2]).astype(np.int32)
```

- (2) Low confidential value when yolov3 prediction

reference:*https://github.com/ultralytics/yolov3/issues/235*

glenn-jocher commented on 4 May 2019 • edited Contributor

@WannaSeaU Ok, I understand what's going on now. If you just have a single class, `CELoss()` notices this, and outputs a zero loss. Since the loss is always zero, the cls output neurons never adjust one way or another, they simply stay the same as their random initialisations. And randomly initialized the model will output a mean around zero with a small standard deviation, which turns into 0.5 after passing through `torch.sigmoid()`. So what we need to do is add an if statement to catch single-class inference, and ignore the `cls` term. I've added this in lines 168-169 in commit

Root cause summary:

(1) single class

→enlarge mini_batch

→shuffle training data order

(2) confine network update on conv_s/m/lbox when first stage epochs

```
__C.TRAIN.FISRT_STAGE_EPOCHS    = 20
```

→let the 2nd stage epochs larger

Summary:

- My recognition model by yolov3 is not good to classify the kangaroo or raccoon, and only the situation that environment near the picture is clean would get a well recognization marker but with cropped one. The root cause would be recognize picture with no convergent yolov3 model. For a big fish, I need to put model training on and on . Otherwise, I need to check why the confidence value is always a low value. The final loss of training(80%) is 3.3886 and test(20%) is 5.9779.