YOLOv4

: Optimal Speed and Accuracy of Object Detection (2020)

2023.03.01

yolov3 -> yolov4

```python
def Load_Yolo_model():
    gpus = tf.config.experimental.list_physical_devices('GPU')
    if len(gpus) > 0:
        print(f'GPUs {gpus}')
        try: tf.config.experimental.set_memory_growth(gpus[0], True)
        except RuntimeError: pass

    if YOLO_FRAMEWORK == "tf": # TensorFlow detection
        if YOLO_TYPE == "yolov4":
            YOLO_V3_WEIGHTS = "model_data/yolov4.weights"
            Darknet_weights = YOLO_V4_WEIGHTS
```

```python
def load_yolo_weights(model, weights_file):
    tf.keras.backend.clear_session() # used to reset layer names
    # load Darknet original weights to TensorFlow model
    if YOLO_TYPE == "yolov4":
        range1 = 110
        range2 = [93, 101, 109]
```

```python
def YOLOv4(input_layer, NUM_CLASS):
    route_1, route_2, conv = cspdarknet53(input_layer)

    route = conv
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = upsample(conv)
    route_2 = convolutional(route_2, (1, 1, 512, 256))
    conv = tf.concat([route_2, conv], axis=-1)
```

```python
def Create_Yolo(input_size=416, channels=3, training=False, CLASSES=YOLO_COCO_CLASSES):
    NUM_CLASS = len(read_class_names(CLASSES))
    input_layer  = Input([input_size, input_size, channels])

    conv_tensors = YOLOv4(input_layer, NUM_CLASS)
```

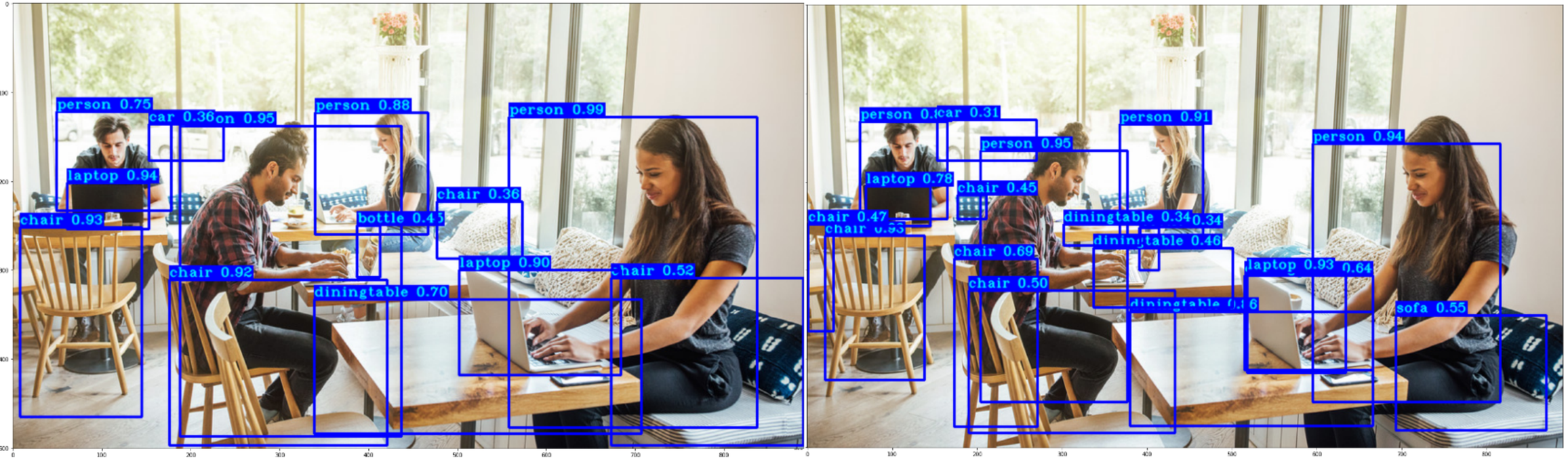
cspdarknet53

```python
def cspdarknet53(input_data):
    input_data = convolutional(input_data, (3, 3,  3,  32), activate_type="mish")
    input_data = convolutional(input_data, (3, 3, 32,  64), downsample=True, activate_type="mish")

    route = input_data
    route = convolutional(route, (1, 1, 64, 64), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 64, 64), activate_type="mish")
    for i in range(1):
        input_data = residual_block(input_data,  64,  32, 64, activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 64, 64), activate_type="mish")

    input_data = tf.concat([input_data, route], axis=-1)

    input_data = convolutional(input_data, (1, 1, 128, 64), activate_type="mish")
    input_data = convolutional(input_data, (3, 3, 64, 128), downsample=True, activate_type="mish")
    route = input_data
    route = convolutional(route, (1, 1, 128, 64), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 128, 64), activate_type="mish")
    for i in range(2):
        input_data = residual_block(input_data, 64,  64, 64, activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 64, 64), activate_type="mish")
    input_data = tf.concat([input_data, route], axis=-1)

    input_data = convolutional(input_data, (1, 1, 128, 128), activate_type="mish")
    input_data = convolutional(input_data, (3, 3, 128, 256), downsample=True, activate_type="mish")
    route = input_data
    route = convolutional(route, (1, 1, 256, 128), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 256, 128), activate_type="mish")
    for i in range(8):
        input_data = residual_block(input_data, 128, 128, 128, activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 128, 128), activate_type="mish")
    input_data = tf.concat([input_data, route], axis=-1)

    input_data = convolutional(input_data, (1, 1, 256, 256), activate_type="mish")
    route_1 = input_data
    input_data = convolutional(input_data, (3, 3, 256, 512), downsample=True, activate_type="mish")
    route = input_data
    route = convolutional(route, (1, 1, 512, 256), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 512, 256), activate_type="mish")
    for i in range(8):
        input_data = residual_block(input_data, 256, 256, 256, activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 256, 256), activate_type="mish")
    input_data = tf.concat([input_data, route], axis=-1)

    input_data = convolutional(input_data, (1, 1, 512, 512), activate_type="mish")
    route_2 = input_data
    input_data = convolutional(input_data, (3, 3, 512, 1024), downsample=True, activate_type="mish")
    route = input_data
    route = convolutional(route, (1, 1, 1024, 512), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 1024, 512), activate_type="mish")
    for i in range(4):
        input_data = residual_block(input_data, 512, 512, 512, activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 512, 512), activate_type="mish")
    input_data = tf.concat([input_data, route], axis=-1)

    input_data = convolutional(input_data, (1, 1, 1024, 1024), activate_type="mish")
    input_data = convolutional(input_data, (1, 1, 1024, 512))
    input_data = convolutional(input_data, (3, 3, 512, 1024))
    input_data = convolutional(input_data, (1, 1, 1024, 512))

    max_pooling_1 = tf.keras.layers.MaxPool2D(pool_size=13, padding='SAME', strides=1)(input_data)
    max_pooling_2 = tf.keras.layers.MaxPool2D(pool_size=9, padding='SAME', strides=1)(input_data)
    max_pooling_3 = tf.keras.layers.MaxPool2D(pool_size=5, padding='SAME', strides=1)(input_data)
    input_data = tf.concat([max_pooling_1, max_pooling_2, max_pooling_3, input_data], axis=-1)

    input_data = convolutional(input_data, (1, 1, 2048, 512))
    input_data = convolutional(input_data, (3, 3, 512, 1024))
    input_data = convolutional(input_data, (1, 1, 1024, 512))

    return route_1, route_2, input_data
```

yolov4

```
def YOLOv4(input_layer, NUM_CLASS):
    route_1, route_2, conv = cspdarknet53(input_layer)

    route = conv
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = upsample(conv)
    route_2 = convolutional(route_2, (1, 1, 512, 256))
    conv = tf.concat([route_2, conv], axis=-1)

    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))

    route_2 = conv
    conv = convolutional(conv, (1, 1, 256, 128))
    conv = upsample(conv)
    route_1 = convolutional(route_1, (1, 1, 256, 128))
    conv = tf.concat([route_1, conv], axis=-1)

    conv = convolutional(conv, (1, 1, 256, 128))
    conv = convolutional(conv, (3, 3, 128, 256))
    conv = convolutional(conv, (1, 1, 256, 128))
    conv = convolutional(conv, (3, 3, 128, 256))
    conv = convolutional(conv, (1, 1, 256, 128))

    route_1 = conv
    conv = convolutional(conv, (3, 3, 128, 256))
    conv_sbbox = convolutional(conv, (1, 1, 256, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    conv = convolutional(route_1, (3, 3, 128, 256), downsample=True)
    conv = tf.concat([conv, route_2], axis=-1)

    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))
    conv = convolutional(conv, (3, 3, 256, 512))
    conv = convolutional(conv, (1, 1, 512, 256))

    route_2 = conv
    conv = convolutional(conv, (3, 3, 256, 512))
    conv_mbbox = convolutional(conv, (1, 1, 512, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    conv = convolutional(route_2, (3, 3, 256, 512), downsample=True)
    conv = tf.concat([conv, route], axis=-1)

    conv = convolutional(conv, (1, 1, 1024, 512))
    conv = convolutional(conv, (3, 3, 512, 1024))
    conv = convolutional(conv, (1, 1, 1024, 512))
    conv = convolutional(conv, (3, 3, 512, 1024))
    conv = convolutional(conv, (1, 1, 1024, 512))

    conv = convolutional(conv, (3, 3, 512, 1024))
    conv_lbbox = convolutional(conv, (1, 1, 1024, 3 * (NUM_CLASS + 5)), activate=False, bn=False)

    return [conv_sbbox, conv_mbbox, conv_lbbox]
```

yolov3 vs yolov4



yolov3 vs yolov4

training for mnist dataset

configs.py

```
# YOLO options
YOLO_TYPE                   = "yolov4" # yolov4 or yolov3
YOLO_FRAMEWORK              = "tf" # "tf" or "trt"
YOLO_V3_WEIGHTS             = "model_data/yolov3.weights"
YOLO_V4_WEIGHTS             = "model_data/yolov4.weights"
YOLO_V3_TINY_WEIGHTS        = "model_data/yolov3-tiny.weights"
YOLO_V4_TINY_WEIGHTS        = "model_data/yolov4-tiny.weights"
YOLO_TRT_QUANTIZE_MODE      = "INT8" # INT8, FP16, FP32
YOLO_CUSTOM_WEIGHTS         = False # "checkpoints/yolov3_custom" #
                            # YOLO_CUSTOM_WEIGHTS also used with Ten
YOLO_COCO_CLASSES           = "model_data/coco/coco.names"
YOLO_STRIDES                = [8, 16, 32]
YOLO_IOU_LOSS_THRESH        = 0.5
YOLO_ANCHOR_PER_SCALE       = 3
YOLO_MAX_BBOX_PER_SCALE     = 100
YOLO_INPUT_SIZE             = 416
if YOLO_TYPE                == "yolov4":
    YOLO_ANCHORS            = [[[12,  16], [19,   36], [40,   28]],
                               [[36,  75], [76,   55], [72,  146]],
                               [[142,110], [192, 243], [459, 401]]]
if YOLO_TYPE                == "yolov3":
    YOLO_ANCHORS            = [[[10,  13], [16,   30], [33,   23]],
                               [[30,  61], [62,   45], [59,  119]],
                               [[116, 90], [156, 198], [373, 326]]]
# Train options
```

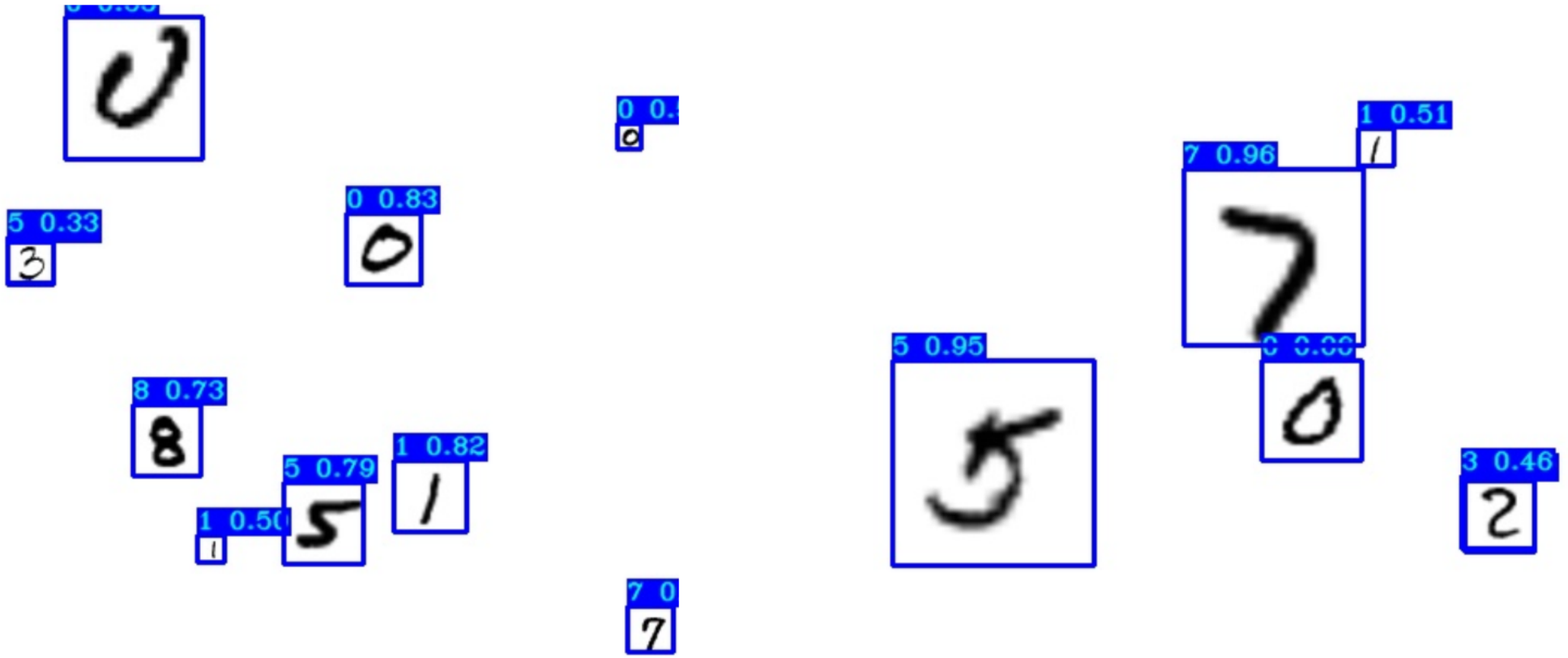
```
cd /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master
```

```
!python mnist/make_data.py
```

```
! python train.py
```

```
! python detect_mnist.py
```

training for mnist dataset



감사합니다 :)

2023.03.01