TensorFlow_2.x_YOLOv3

2023.02.08

TensorFlow_2.x_YOLOv3

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

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

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

```
Extracting all train files now...
Done!
Extracting all test files now...
Done!
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000001.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000002.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000003.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000004.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000005.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000006.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000007.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000008.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000009.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000010.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000011.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000012.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000013.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000014.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000015.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000016.jpg
=> /content/drive/MyDrive/TensorFlow-2.x-YOLOv3-master/mnist/mnist_train/000017.jpg
```

TensorFlow_2.x_YOLOv3

```
#Training
!python train.py
```

```
epoch:29 step:  213/250, lr:0.000001, giou_loss:   2.72, conf_loss:   0.35, prob_loss:   2.83, total_loss:   5.90
epoch:29 step:  214/250, lr:0.000001, giou_loss:   2.42, conf_loss:   0.31, prob_loss:   4.77, total_loss:   7.50
epoch:29 step:  215/250, lr:0.000001, giou_loss:   1.89, conf_loss:   0.40, prob_loss:   2.12, total_loss:   4.41
epoch:29 step:  216/250, lr:0.000001, giou_loss:   2.29, conf_loss:   0.33, prob_loss:   1.89, total_loss:   4.51
epoch:29 step:  217/250, lr:0.000001, giou_loss:   1.98, conf_loss:   0.34, prob_loss:   2.06, total_loss:   4.37
epoch:29 step:  218/250, lr:0.000001, giou_loss:   3.12, conf_loss:   0.52, prob_loss:   2.73, total_loss:   6.36
epoch:29 step:  219/250, lr:0.000001, giou_loss:   2.20, conf_loss:   0.46, prob_loss:   2.03, total_loss:   4.69
epoch:29 step:  220/250, lr:0.000001, giou_loss:   2.55, conf_loss:   0.64, prob_loss:   3.08, total_loss:   6.27
epoch:29 step:  221/250, lr:0.000001, giou_loss:   2.16, conf_loss:   0.29, prob_loss:   2.03, total_loss:   4.49
epoch:29 step:  222/250, lr:0.000001, giou_loss:   3.12, conf_loss:   0.33, prob_loss:   2.75, total_loss:   6.21
epoch:29 step:  223/250, lr:0.000001, giou_loss:   2.29, conf_loss:   0.31, prob_loss:   1.91, total_loss:   4.51
epoch:29 step:  224/250, lr:0.000001, giou_loss:   2.82, conf_loss:   0.43, prob_loss:   2.54, total_loss:   5.79
epoch:29 step:  225/250, lr:0.000001, giou_loss:   2.26, conf_loss:   0.36, prob_loss:   2.43, total_loss:   5.05
epoch:29 step:  226/250, lr:0.000001, giou_loss:   2.41, conf_loss:   0.53, prob_loss:   1.97, total_loss:   4.90
epoch:29 step:  227/250, lr:0.000001, giou_loss:   4.54, conf_loss:   3.01, prob_loss:   5.30, total_loss:  12.86
epoch:29 step:  228/250, lr:0.000001, giou_loss:   1.84, conf_loss:   0.42, prob_loss:   2.03, total_loss:   4.30
epoch:29 step:  229/250, lr:0.000001, giou_loss:   1.54, conf_loss:   0.29, prob_loss:   1.38, total_loss:   3.21
epoch:29 step:  230/250, lr:0.000001, giou_loss:   1.46, conf_loss:   0.33, prob_loss:   1.53, total_loss:   3.33
epoch:29 step:  231/250, lr:0.000001, giou_loss:   3.10, conf_loss:   1.85, prob_loss:   3.61, total_loss:   8.56
epoch:29 step:  232/250, lr:0.000001, giou_loss:   2.92, conf_loss:   0.41, prob_loss:   2.94, total_loss:   6.27
epoch:29 step:  233/250, lr:0.000001, giou_loss:   1.83, conf_loss:   0.30, prob_loss:   2.02, total_loss:   4.15
epoch:29 step:  234/250, lr:0.000001, giou_loss:   0.87, conf_loss:   0.27, prob_loss:   0.86, total_loss:   2.00
epoch:29 step:  235/250, lr:0.000001, giou_loss:   3.57, conf_loss:   0.64, prob_loss:   3.63, total_loss:   7.83
epoch:29 step:  236/250, lr:0.000001, giou_loss:   3.74, conf_loss:   2.48, prob_loss:   4.67, total_loss:  10.90
epoch:29 step:  237/250, lr:0.000001, giou_loss:   3.19, conf_loss:   1.91, prob_loss:   3.52, total_loss:   8.63
epoch:29 step:  238/250, lr:0.000001, giou_loss:   2.55, conf_loss:   0.51, prob_loss:   2.92, total_loss:   5.98
epoch:29 step:  239/250, lr:0.000001, giou_loss:   2.82, conf_loss:   0.38, prob_loss:   2.87, total_loss:   6.06
epoch:29 step:  240/250, lr:0.000001, giou_loss:   2.53, conf_loss:   0.48, prob_loss:   2.43, total_loss:   5.43
epoch:29 step:  241/250, lr:0.000001, giou_loss:   2.22, conf_loss:   0.37, prob_loss:   1.78, total_loss:   4.36
epoch:29 step:  242/250, lr:0.000001, giou_loss:   2.44, conf_loss:   0.39, prob_loss:   2.30, total_loss:   5.13
epoch:29 step:  243/250, lr:0.000001, giou_loss:   2.23, conf_loss:   0.47, prob_loss:   1.93, total_loss:   4.63
epoch:29 step:  244/250, lr:0.000001, giou_loss:   2.06, conf_loss:   0.32, prob_loss:   2.08, total_loss:   4.46
epoch:29 step:  245/250, lr:0.000001, giou_loss:   2.39, conf_loss:   0.30, prob_loss:   2.44, total_loss:   5.13
epoch:29 step:  246/250, lr:0.000001, giou_loss:   2.38, conf_loss:   0.58, prob_loss:   2.42, total_loss:   5.38
epoch:29 step:  247/250, lr:0.000001, giou_loss:   2.39, conf_loss:   0.36, prob_loss:   2.67, total_loss:   5.42
epoch:29 step:  248/250, lr:0.000001, giou_loss:   3.68, conf_loss:   1.29, prob_loss:   3.21, total_loss:   8.18
epoch:29 step:  249/250, lr:0.000001, giou_loss:   2.50, conf_loss:   0.35, prob_loss:   2.45, total_loss:   5.30
epoch:29 step:    0/250, lr:0.000001, giou_loss:   2.11, conf_loss:   0.31, prob_loss:   1.80, total_loss:   4.22
epoch:29 step:    1/250, lr:0.000001, giou_loss:   3.21, conf_loss:   0.38, prob_loss:   2.94, total_loss:   6.52


giou_val_loss:   2.48, conf_val_loss:   0.63, prob_val_loss:   4.27, total_val_loss:   7.38
```

configs.py

```
    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
TRAIN_YOLO_TINY             = False
TRAIN_SAVE_BEST_ONLY        = True # saves only best model according validation
TRAIN_SAVE_CHECKPOINT       = False # saves all best validated checkpoints in t
TRAIN_CLASSES               = "mnist/mnist.names"
TRAIN_ANNOT_PATH            = "mnist/mnist_train.txt"
TRAIN_LOGDIR                = "log"
TRAIN_CHECKPOINTS_FOLDER    = "checkpoints"
TRAIN_MODEL_NAME            = f"{YOLO_TYPE}_custom"
TRAIN_LOAD_IMAGES_TO_RAM    = True # With True faster training, but need more R
TRAIN_BATCH_SIZE            = 4
TRAIN_INPUT_SIZE            = 416
TRAIN_DATA_AUG              = True
TRAIN_TRANSFER              = True
TRAIN_FROM_CHECKPOINT       = False # "checkpoints/yolov3_custom"
TRAIN_LR_INIT               = 1e-4
TRAIN_LR_END                = 1e-6
TRAIN_WARMUP_EPOCHS         = 2
TRAIN_EPOCHS                = 30
```

TensorFlow_2.x_YOLOv3

- train.py

  ⇒ main() ∨

```
trainset = Dataset('train')
testset = Dataset('test')

steps_per_epoch = len(trainset)
global_steps = tf.Variable(1, trainable=False, dtype=tf.int64)
warmup_steps = TRAIN_WARMUP_EPOCHS * steps_per_epoch
total_steps = TRAIN_EPOCHS * steps_per_epoch

if TRAIN_TRANSFER:
    Darknet = Create_Yolo(input_size=YOLO_INPUT_SIZE, CLASSES=YOLO_COCO_CLASSES)
    load_yolo_weights(Darknet, Darknet_weights) # use darknet weights

yolo = Create_Yolo(input_size=YOLO_INPUT_SIZE, training=True, CLASSES=TRAIN_CLASSES)

if TRAIN_FROM_CHECKPOINT:
    try:
        yolo.load_weights(f"./checkpoints/{TRAIN_MODEL_NAME}")
    except ValueError:
        print("Shapes are incompatible, transfering Darknet weights")
        TRAIN_FROM_CHECKPOINT = False

if TRAIN_TRANSFER and not TRAIN_FROM_CHECKPOINT:

    for i, l in enumerate(Darknet.layers):
        layer_weights = l.get_weights()
        if layer_weights != []:
            try:

                yolo.layers[i].set_weights(layer_weights)
            except:
                print("skipping", yolo.layers[i].name)

optimizer = tf.keras.optimizers.Adam()
```

TensorFlow_2.x_YOLOv3

- train.py

  ⇒ main()

  ⇒ train_step() v

```python
def train_step(image_data, target):
    with tf.GradientTape() as tape:
        pred_result = yolo(image_data, training=True)
        giou_loss=conf_loss=prob_loss=0

        # optimizing process
        grid = 3 if not TRAIN_YOLO_TINY else 2
        for i in range(grid):
            conv, pred = pred_result[i*2], pred_result[i*2+1]
            loss_items = compute_loss(pred, conv, *target[i], i, CLASSES=TRAIN_CLASSES)
            giou_loss += loss_items[0]
            conf_loss += loss_items[1]
            prob_loss += loss_items[2]

        total_loss = giou_loss + conf_loss + prob_loss

        gradients = tape.gradient(total_loss, yolo.trainable_variables)
        optimizer.apply_gradients(zip(gradients, yolo.trainable_variables))

        # update learning rate
        # about warmup: https://arxiv.org/pdf/1812.01187.pdf&usg=ALkJrhglKOPDjNt6SHGbphTHyMcT0cuMJg
        global_steps.assign_add(1)
        if global_steps < warmup_steps:# and not TRAIN_TRANSFER:
            lr = global_steps / warmup_steps * TRAIN_LR_INIT
        else:
            lr = TRAIN_LR_END + 0.5 * (TRAIN_LR_INIT - TRAIN_LR_END)*(
                (1 + tf.cos((global_steps - warmup_steps) / (total_steps - warmup_steps) * np.pi)))
        optimizer.lr.assign(lr.numpy())

        # writing summary data
        with writer.as_default():
            tf.summary.scalar("lr", optimizer.lr, step=global_steps)
            tf.summary.scalar("loss/total_loss", total_loss, step=global_steps)
            tf.summary.scalar("loss/giou_loss", giou_loss, step=global_steps)
            tf.summary.scalar("loss/conf_loss", conf_loss, step=global_steps)
            tf.summary.scalar("loss/prob_loss", prob_loss, step=global_steps)
        writer.flush()

    return global_steps.numpy(), optimizer.lr.numpy(), giou_loss.numpy(), conf_loss.numpy(), prob_loss.numpy(), total_loss.numpy()
```

TensorFlow_2.x_YOLOv3

- train.py
  ⇒ main()
  ⇒ train_step() v

```
def validate_step(image_data, target):
    with tf.GradientTape() as tape:
        pred_result = yolo(image_data, training=False)
        giou_loss=conf_loss=prob_loss=0

        # optimizing process
        grid = 3 if not TRAIN_YOLO_TINY else 2
        for i in range(grid):
            conv, pred = pred_result[i*2], pred_result[i*2+1]
            loss_items = compute_loss(pred, conv, *target[i], i, CLASSES=TRAIN_CLASSES)
            giou_loss += loss_items[0]
            conf_loss += loss_items[1]
            prob_loss += loss_items[2]

        total_loss = giou_loss + conf_loss + prob_loss

    return giou_loss.numpy(), conf_loss.numpy(), prob_loss.numpy(), total_loss.numpy()
```

TensorFlow_2.x_YOLOv3

```python
def compute_loss(pred, conv, label, bboxes, i=0, CLASSES=YOLO_COCO_CLASSES):
    NUM_CLASS = len(read_class_names(CLASSES))
    conv_shape  = tf.shape(conv)
    batch_size  = conv_shape[0]
    output_size = conv_shape[1]
    input_size  = STRIDES[i] * output_size
    conv = tf.reshape(conv, (batch_size, output_size, output_size, 3, 5 + NUM_CLASS))

    conv_raw_conf = conv[:, :, :, :, 4:5]
    conv_raw_prob = conv[:, :, :, :, 5:]

    pred_xywh     = pred[:, :, :, :, 0:4]
    pred_conf     = pred[:, :, :, :, 4:5]

    label_xywh    = label[:, :, :, :, 0:4]
    respond_bbox  = label[:, :, :, :, 4:5]
    label_prob    = label[:, :, :, :, 5:]

    giou = tf.expand_dims(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 = bbox_iou(pred_xywh[:, :, :, :, np.newaxis, :], bboxes[:, np.newaxis, np.newaxis, np.newaxis, :, :])
    # Find the value of IoU with the real box The largest prediction box
    max_iou = tf.expand_dims(tf.reduce_max(iou, axis=-1), axis=-1)

    # If the largest iou is less than the threshold, it is considered that the prediction box contains no objects, then the background box
    respond_bgd = (1.0 - respond_bbox) * tf.cast( max_iou < YOLO_IOU_LOSS_THRESH, tf.float32 )

    conf_focal = tf.pow(respond_bbox - pred_conf, 2)

    # Calculate the loss of confidence
    # we hope that if the grid contains objects, then the network output prediction box has a confidence of 1 and 0 when there is no object.
    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]))

    return giou_loss, conf_loss, prob_loss
```

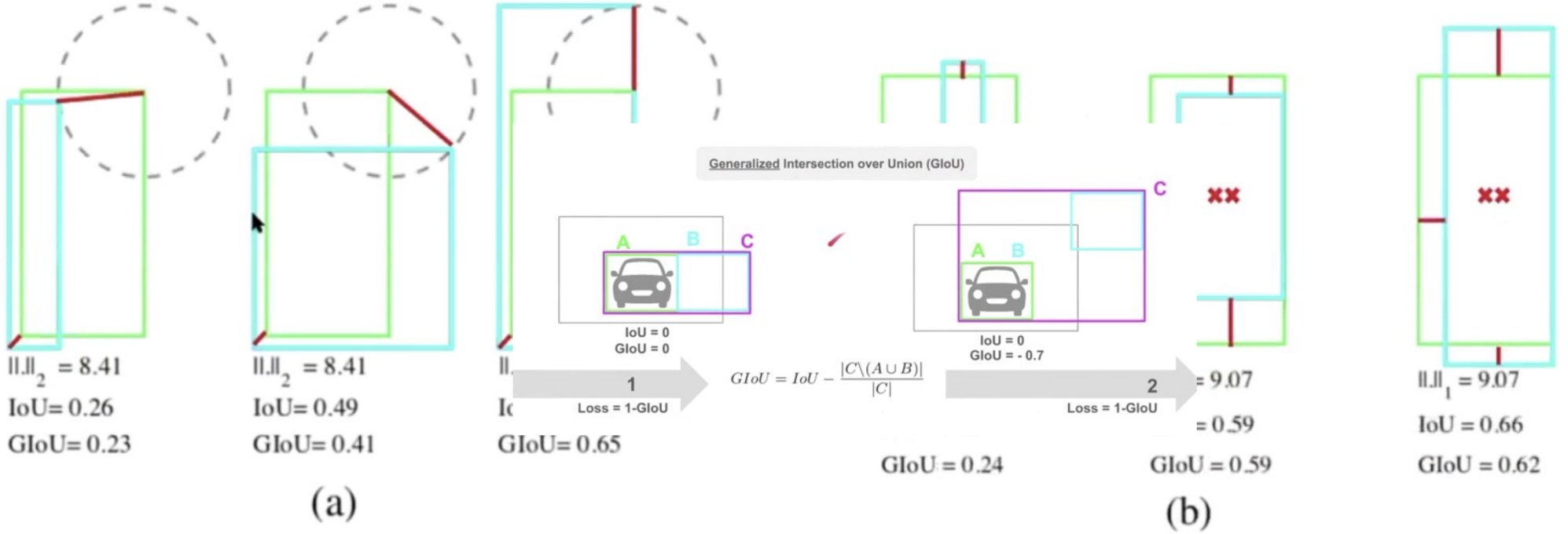
- train.py

  ⇒ main()

  ⇒ train_step() v

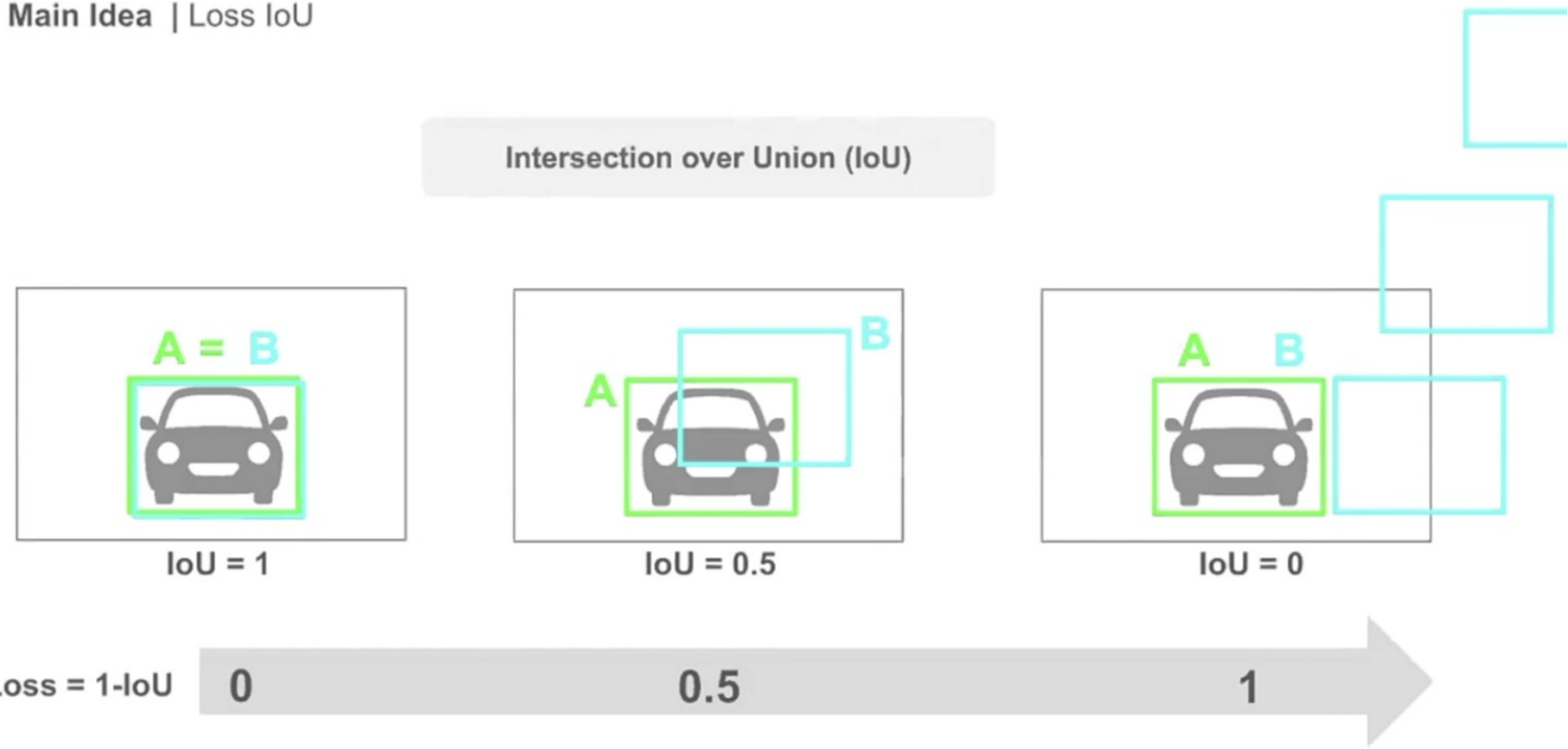
  ⇒ compute_loss() v

GIoU loss란?


Generalized Intersection over Union (GIoU)
A
B
C
IoU = 0
GIoU = 0
1
Loss = 1-GIoU
$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$
IoU = 0
GIoU = - 0.7
2
Loss = 1-GIoU
$\|.\|_2$ = 8.41
IoU= 0.26
GIoU= 0.23
$\|.\|_2$ = 8.41
IoU= 0.49
GIoU= 0.41
GIoU= 0.65
GIoU = 0.24
= 9.07
= 0.59
GIoU = 0.59
$\|.\|_1$ = 9.07
IoU = 0.66
GIoU = 0.62
(a)
(b)

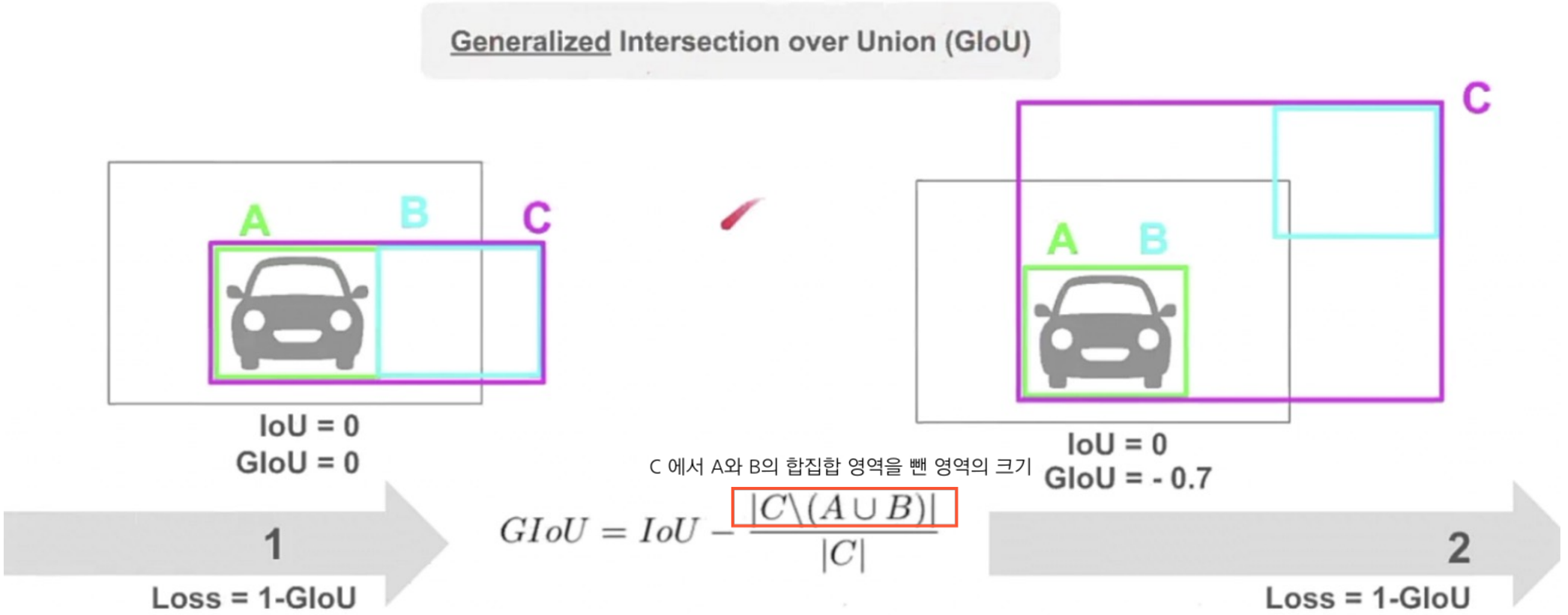
GIoU loss란?

Main Idea | Loss IoU



- IoU를 Loss로 사용하려면 1 – IoU를 사용하여 두 박스가 잘 겹칠수록 0에 가까워지도록 한다.
  -> But, 세번째 경우에서 문제 발생! 두 박스가 겹치지 않을 경우는?

GIoU loss란?

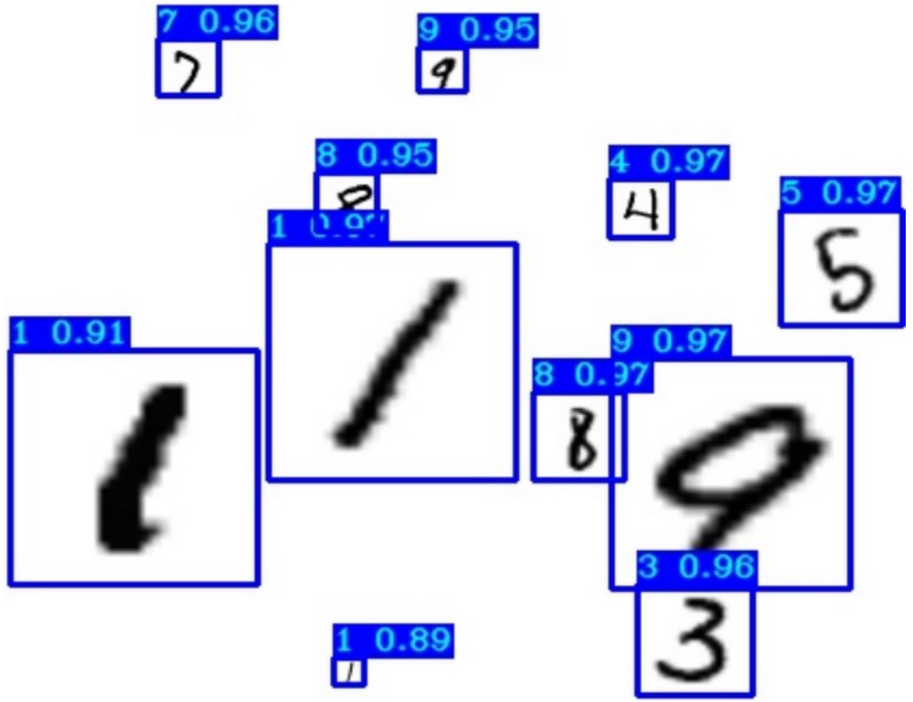


- GIoU를 Loss로 사용할 때에는 Loss = 1 – GIoU 형태로 사용하며 Loss의 최대값은 2, 최소값은 0이 되도록 설정

TensorFlow_2.x_YOLOv3

```
1 !python detect_mnist.py
```

```
2023-02-07 18:47:40.143938: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:42]
```



감사합니다 :)

2023.02.08