An interesting exploration of GIOU loss and YOLO v3 model

1st Yuan Gao
Faculty of Engineering
University of Ottawa
Ottawa, Canada
ygao133@uottawa.ca

2nd Ao Peng
Faculty of Engineering
University of Ottawa
Ottawa, Canada
apeng055@uottawa.ca

***Abstract*—We choose [1] as our main topic paper and the authors propose a generalized version of *IoU* (*GIoU*) in order to solve the main weakness of *IoU*. To figure out whether *GIoU* can work well or not and how it will work, we start our research following this paper and compare three different loss functions which are localization loss, *IoU* loss and *GIoU* loss based on YOLO v3. In addition, we do some experiments on focal loss and propose 2 ideas on YOLO v3 with *GIoU* loss model, which we called asymptotic fine tuning strategy and prediction layer separation. Code has been made publicly available at https://github.com/bitgs/YOLO_GIOU.**

***Index Terms*—GIoU, IoU, YOLO v3, Focal loss, asymptotic fine tuning**

I. Introduction

In computer vision area, Intersection over Union (*IoU*) is the most famous way as the evaluation metric used in the object detection benchmarks for comparing the similarity between two arbitrary shapes. In general, we use *IoU* based on the widths, heights and locations of two bounding boxes as the region property and then calculate a normalized value that represents the overlapping area of these bounding boxes. However, there is a major problem that it cannot work well when it faces Non-overlapping cases Fig.1. In order to solve this problem, [1] proposed a generalized version of *IoU* named Generalized Intersection of Union (*GIoU*). This method not only rectifies the weakness of *IoU* but also keeps the property of *IoU* so that it can be generally used in object detection area.

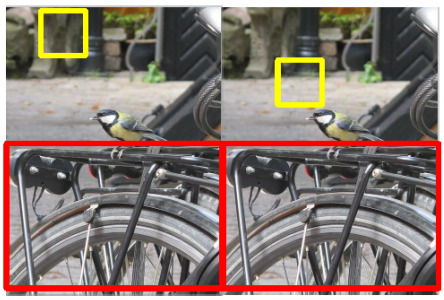

Fig. 1: Non-overlapping cases. In theses cases, The *IoU* value still be 0 [1].

In this report, we will introduce the weakness of *IoU* by using non-overlapping cases and explain the definition of *GIoU* and how it works in that situation. We use these two methods as the loss function into YOLO v3 [2] to find correlations and differences between them based on PASCAL VOC 2012 trainval and VOC 2007 trainval and test dataset and test on VOC 2012 dataset. In addition, we explore the use of focal loss and propose some new ideas based on *GIoU*. All details are shown in section IV part E and F.

The main content of this report is summarized as follows:

- We introduce the architecture of YOLO v3.
- We introduce *IoU* and *GIoU* and use them with localization loss [2] as the loss functions of YOLO v3 to test which of them can get the best performance.
- We do an exploration about focal loss.
- We propose two ideas which are the strategy of asymptotic fine tuning and the assumption of separating the branch of different loss functions.

II. YOLO v3

A. Architecture

YOLO v3 is one of state-of-the art object detection algorithms. The architecture of YOLO v3 is a combination of three parts which are from ResNet [3], Feature Pyramid Networks (FPN) [4] and Single Shot MultiBox Detector (SSD) [5] respectively. From Fig.2, we can see the main structure of YOLO v3 is based on SSD. There is a difference in the backbone where the authors use residual units to extract features. The output are three different feature maps corresponding three bounding boxes with different size for specific detection task which are based on FPN.

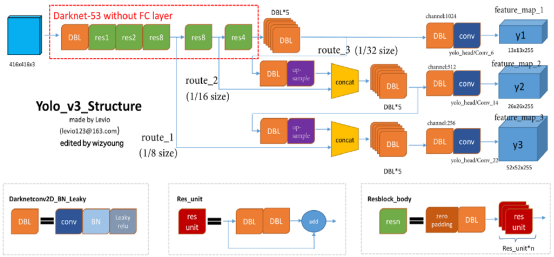

Fig. 2: The whole architecture of YOLO v3 [6].

B. Single Shot MultiBox Detector

The base architecture is from SSD and the authors of [2] follow its design with some modifies. This part is based on a feed-forward convolutional network which produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in the boxes [5]. However, the improvements of backbone make YOLO v3 can get better performance.

The main modifies between SSD and YOLO v3 is summarized as follows:

- For the backbone, SSD uses VGG19 while YOLO v3 uses Darknet-53.
- YOLO v3 changes the loss function from softmax to logistic regression.
- YOLO v3 uses FPN structure

C. Residual Structure

Vanishing gradient is a serious dilemma in deep network and it is this structure that supports deeper architecture of networks of YOLO v3 which prevents that problem. The residual units [Fig.2] which are used in backbone of YOLO v3 need less parameters and simplify the process of calculation.

D. Feature Pyramid Network

There are many objects with different sizes and corresponding features. Based on that, we can use the feature in shallow layers to recognize the objects and for complex targets we can use deep layer features Fig.3. This is another main difference between SSD and YOLO v3.

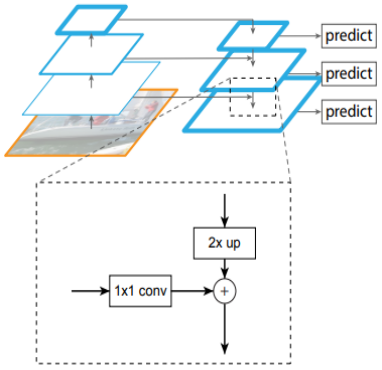

Fig. 3: The FPN structure [4] is corresponding the head of YOLO v3 in Fig.1.

E. Loss Function

The loss function of YOLO v3 is split into 3 parts and each part has a specific purpose in (1).

$$Loss_{total} = Loss_{objectness} + Loss_{classification} + Loss_{localization} \tag{1}$$

$Loss_{objectness}$ is to detect whether there is an object or not; $Loss_{classification}$ is the function for classification task which is to judge which class this object belongs to; Last but not least, $Loss_{localization}$ is to calculate the distance between predictions and ground truth.

However, authors [1] replace the $Loss_{localization}$ to $Loss_{IoU}$ as there is no obvious correlation between minimizing the localization loss and improve its *IoU* value in Fig.4.

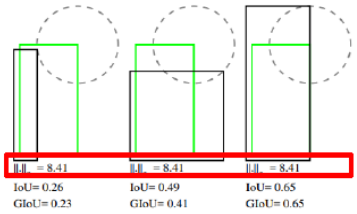

Fig. 4: The $L_2 - norm$ is still 0 in 3 different situations while $Loss_{IoU}$ can reflect the third one can get best performance [1].

III. Generalized IoU

In order to to solve the weakness of *IoU*, the authors [1] proposed *GIoU*.

We first need to find the smallest area which includes both A and B (see Fig. 5). Then we calculate a ratio between the area of C excluding A and B and the total area of C. Finally we can get the *GIoU* by subtracting this ratio from the IoU.

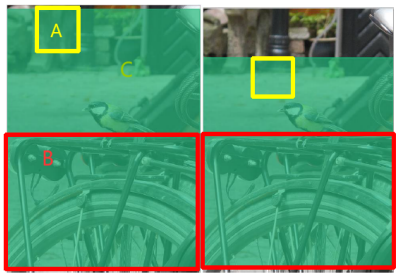

Fig. 5: GIoU works well on these 2 situations [1].

The calculation of *GIoU* is summarized as below:

$$\begin{aligned} IoU &= \frac{|A \cap B|}{|A \cup B|} \\ GIoU &= IoU - \frac{|C \setminus (A \cup B)|}{|C|} \end{aligned} \tag{2}$$

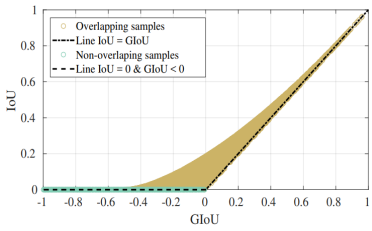

Fig. 6: Correlation between *GIoU* and *IoU* for overlapping and non-overlapping samples [1].

Compared to *IoU*, the major improvement of *GIoU* is that it can deal with non-overlapping cases and keeps the properties of *IoU*. From Fig. 6, when the value of *GIoU* asymptotically

converges to 1, *GIoU* curve is becoming to overlap the *IoU* curve which means *GIoU* is becoming to *IoU*, which supports that *GIoU* still has the properties of *IoU*; nevertheless, when the overlapping area is 0, *IoU* has no value and therefore no gradient. However, *GIoU* is always differentiable.

IV. Project Works

In this project, we first explored the ideas and implementation details by paper reading and source code reading. Then we learned the training and evaluating process on benchmarks and the works of data augmentation. After this learning, we trained the first YOLO v3 model and reproduced the experimental results. Due to the limitation of computing resources, we only trained the model on PASCAL VOC benchmark. After the first model trained, we followed the idea of *GIoU*, and converted $Loss_{localization}$ of YOLOv3 to *IoU* and *GIoU* loss respectively. At last, we tried to absorb some ideas by others and proposed some ideas to see if these can further improve the average precision(AP).

A. Data Augmentation

To make the model more robust to various input object sizes and shapes, following data augmentation methods are applied:

- Multi-scale Training: image sizes vary from [320, 320] to [640, 640] at an interval of 32. Change the scale of images every 10 batches.
- Crop an image randomly with bounding box constraints. This data augmentation is used in training of Single Shot Multibox Detector [5]. More details can be found in data augmentation section of the original paper.

After the aforementioned sampling step, each sampled patch is resized to fixed size and is horizontally flipped with probability of 0.5, in addition to applying some photo-metric distortions similar to those described in [7].

B. Training Details and Baseline Model

We use the Tensorflow versioned off-the-shelf YOLO v3 model by [6]. We use a two-stage training strategy. First stage: Restore the backbone (Darknet-53 without FC layer) weights from the checkpoint, which is trained on Imagenet ILSVRC 2012 classification dataset. Then we fix the backbone part and train the YOLO v3 head with learning rate 0.001 until the loss reaches to a low level plateau. Second stage: Restore the weights from the first stage, then train the whole model with learning rate 0.0001 for 30 epochs and 0.00001 for another 10 epochs. We train the model using SGD, 0.99 momentum, 0.0005 weight decay and batch size 4. We train the YOLOv3 model on PASCAL VOC2012 trainval and VOC2007 trainval and test dataset and test on VOC2012 test. The main performance measure used in this benchmark is shown by AP, which is averaging mAP across different value of *IoU* thresholds, i.e. *IoU* = .5, .55, $\cdots$, .95. This model achieves 0.8246 AP, see table 1. We use the model as baseline.

C. IOU and GIOU Model

When training the IOU and GIOU model, we change the localization loss (3) of YOLOv3 to IOU loss and GIOU loss respectively. The localization loss is a Smooth L1 loss [8] between the predicted box (l) and the ground truth box (g) parameters. e regress to offsets for the center (cx, cy) of the default bounding box (d) and for its width (w) and height (h).

$$
\begin{aligned}
L_{loc}(x,l,g) &= \sum_{i\in Pos}^{N}\sum m \in \{cx,cy,w,h\} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m) \\
\hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^w \\
\hat{g}_j^{cy} &= (g_j^{cy} - d_i^{cy})/d_i^h \\
\hat{g}_j^{w} &= log(\frac{g_j^w}{d_i^w}) \\
\hat{g}_j^{h} &= log(\frac{g_j^h}{d_i^h})
\end{aligned}
\tag{3}
$$

We keep the objectiveness loss and classification loss as described in the paper (see (4)).

$$
Loss_{total} = Loss_{objectness} + Loss_{classification} + Loss_{IoU/GIoU} \tag{4}
$$

By reading the C++ source code of the author, we know that only positive samples (the anchor with best *IoU*) contribute to *IoU* loss and *GIoU* loss. We follow the same training method as described above. We plot the result in TABLE I. The result shows a 0. 52 points of mAP increase by IOU model and a 0. 65 points increase by *GIoU* model.

TABLE I: AP trained on PASCAL VOC2012 trainval and VOC2007 trainval and test dataset and tested on VOC2012 test.

	$Loss_{localization}$	$Loss_{IoU}$	$Loss_{GIoU}$
AP	0.8226	0.8298	0.8311

D. Focal Loss

The author of YOLO v3 said the use of Focal loss [9] dropped the mAP about 2 points. We think the possible reason is that the original threshold of 0.5 is too high for the specific model. Focal loss is used to alleviate the problem of class imbalance between positive and negative samples. Followed the thought by the author we think the reason why Focal loss does not help lies in 3 reasons:

- Positive samples contribute to (1)objectiveness loss, (2)classification loss and (3)localization loss.
- Negative samples contribute to only (1)objectiveness.
- Only samples with IOU$<$0.5 are treated as negative samples.

Therefore the class imbalance problem should not be severe in YOLOv3 model. However, the real case is that the number of positive samples is less than 5 and by contrast the number of negative samples is always more than 10,000. So, theoretically, Focal loss should help.

We use red indicates ground truth, yellow indicates negative and blue indicates ignored in Fig.7. We think the reason might

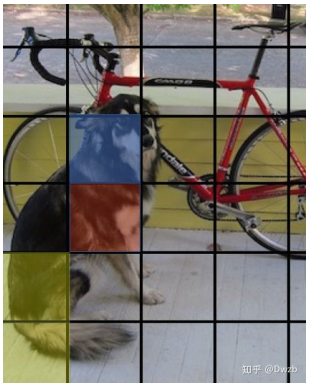

Fig. 7: Demo image of positive samples, negative samples and ignored samples.

be the threshold 0.5 is too high, which results in the hard negative samples, which are shown in yellow, look really like positive samples, the red one. Hence, the learning process is confused that whether these fluffy parts belong to a dog. We test the influence of the threshold to the AP and plot the result in TABLE II. The use of Focal loss improves the AP about 2 points when using the same threshold of 0.5 because of the significant decrease in the objectiveness loss. Lower the threshold to 0.45 gives us the best result, a 2.2-2.4 points AP increase.

TABLE II: Comparison between models not using Focal loss and using Focal loss with different threshold for negative samples.

	$Loss_{localization}$	$Loss_{IoU}$	$Loss_{GIoU}$
w/o FL(threshold 0.5)	0.8426	0.8298	0.8311
w/ FL(threshold 0.4)	0.8342	0.8396	0.8409
w/ FL(threshold 0.45)	**0.8482**	**0.8521**	**0.8536**
w/ FL(threshold 0.5)	0.8455	0.8501	0.8512

E. Asymptotic Fine Tuning Strategy

The typical training method for detection works are first pre-training a backbone network on Imagenet dataset and then fine-tuning the detection head on the object detection dataset without the fully connected or average pooling layer of the backbone network. We noticed that the image space or the feature patterns of the Imagenet dataset are always different from the object detection dataset. So we add one more training step in between these 2 steps. Imagenet is an image classification challenge with 1000 classes, so the final activation function is always softmax and the number of output is 1000. Before training on object detection tasks, we first convert the object detection task into a multi-class classification task, see Fig.8.

We change the number of the output of the last layer and replace the final activation function to a sigmoid function. We call this step as an asymptotic training step and we hope after this step the backbone can have more related feature information.

After that we train the detection head as usual. This fine training method increases the AP about 0.9 points.

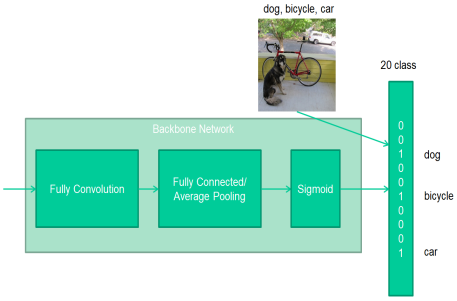

Fig. 8: Demo of Asymptotic fine tuning strategy.

TABLE III: Comparison between models using and not using asymptotic fine tuning strategy.

	$Loss_{localization}$	$Loss_{IoU}$	$Loss_{GIoU}$
w/o asymptotic fine tuning	0.8482	0.8521	0.8536
w/ asymptotic fine tuning	0.8576	0.8602	0.8613

F. Prediction Layer Separation

A typical prediction layer is shown in the top half of Fig.9, the number of channels is equal to 1 plus the number of classes predicted plus 4. The first 1 is used to predict whether the anchor has an object and the last four are used to predict the offset of the x, y, w and h.

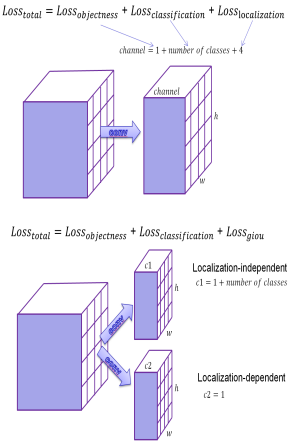

Fig. 9: Demo of prediction layer separation.

We think that the first 2 parts are localization uncorrelated and the last part is localization correlated, so it might be helpful to separate these 2 kinds of prediction into 2 separated convolution branches (bottom half of Fig.9). The first branch is used to predict what it is and the second branch is used to predict where it is.

TABLE IV: Comparison between models using and not using asymptotic fine tuning strategy.

	$Loss_{GIoU}$
w/o prediction layer separation	0.8613
w/ prediction layer separation	0.8605

However, the result (TABLE IV) does not show an improvement in the performance. We think the reason might be that these two kinds of tasks better utilize different level of features and the performance might be improved by moving the separation a bit more forward. We leave this as our future work.

V. Conclusion

In this paper, we first compared the performance of *IoU* and *GIoU* loss with the $Loss_{localization}$ used in YOLO v3. We showed that these two losses can increase the AP by 0.52 and 0.65 points in YOLO v3 model. We also used the Focal loss with different threshold and showed that Focal loss can significantly increase the performance by 2 points, and by lowering the threshold to 0.45 the performance can be further improved by 0.2-0.4 points. At last, we proposed the ideas of asymptotic-fine-tuning strategy and prediction layer separation. The former one shows a 0.9 points AP rise with the idea of narrowing the gap between pre-training dataset and object-detection dataset, while the latter one does not show an improvement in performance and we leave that as our future works.

References

[1] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union," June 2019.

[2] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, June 2016.

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[6] "Yolov3 tensorflow implementation." [Online]. Available: https://github.com/wizyoung/YOLOv3_TensorFlow

[7] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[8] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[9] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.