Reproducibility Study of "Scaled-YOLOv4: Scaling Cross Stage Partial Network"

Tianyu Gu[1] and Xinyu Liu[2]
1. *Department of Civil Engineering, EPFL, Switzerland*
2. *Department of Electrical Engineering, EPFL, Switzerland*

Summary

1) *Scope of Reproducibility*

This work attempts to reproduce the results of the 2021 CVPR paper, Scaled-YOLOv4: Scaling Cross Stage Partial Network. As the algorithm uses plenty of tricks to improve detection accuracy, not only the main claims but also several claims regarding details of the algorithm are examined.

2) *Methodology*

The code that the authors published alongside their paper is used to understand the methodology of their experiments, which is only briefly touched upon in the original paper. The contribution of this study is to extrapolate the original method using the provided code and to use this to try to recreate the experiments. Code is adapted for different testing purposes.

3) *Results*

The experiment results of this study follows similar patterns as those of the authors generally. However, some results slightly deviate from the authors' claimed results, meaning the original paper has some reproducibility issues.

4) *What was difficult*

The original paper doesn't include many details so we read a lot of Q&As in the author's github, trying to understand the parameters they used for tuning the ML model. Also, the authors build their algorithms based on MS COCO dataset, which is a large-size dataset requiring very high time expense. One of our teammates dropped out suddenly in this project, so it is really hard for us two to use the limited computation resource to complete this task.

I. Introduction

Object detection is a core problem in the field of computer vision, and its main purpose is to find particular objects in images or videos, while detecting their locations, sizes and categories. Object detection has broad application prospects, therefore it has become a research hotspot in recent years. Convolutional neural network (CNN) based detectors are proved to have high inference speed and great feature extraction capability, and they can be separated into two categories — one-stage and two-stage methods. One-stage detectors, such as YOLO series models [1]–[5] and SSD series models [6], [7], generally have high inference speeds while, two-stage detectors, such as R-CNN series models [8]–[10], FPN [11] have high localization and recognition accuracy.

In 2021, a set of object detection algorithms named scaled-YOLOv4 [5], were proposed based on YOLOv4 and cross-stage-partial (CSP) approach. The scaled-YOLOv4 algorithms can be divided into three groups: YOLOv4-tiny, YOLOv4-CSP and YOLOv4-large, applicable to small, medium and large networks for low-end, general and high-end GPUs, respectively. The authors claim that YOLOv4-large achieves the highest accuracy on MS COCO dataset compared with any other object detection algorithms, and YOLOv4-CSP has better speed/accuracy trade-off compared with original YOLOv4, and finally YOLOv4-tiny has the fastest inference speed and best accuracy in comparison with other tiny models.

Except for these core claims, the authors also made several claims regarding details of the algorithms [4], [5]. They claim that k-means is used to get the anchor box sizes in order to achieve the best convergence speed, and they compared different loss and activation functions, finally selected CIoU loss and Mish activation function for their superior performance.

This paper investigates the reproducibility of the scaled-YOLOv4 research, and the above claims are tested, analyzed, and evaluated. In addition, the authors derived their results based on COCO dataset, and in this paper VOC dataset is also tested to see if the results have good generalization effect.

II. Scope of reproducibility

Focus of this paper is to reproduce the general trends in the authors' experimental results on MS COCO dataset and test if there are similar trends on other datasets. Several claims in the scaled-YOLOv4 paper are,

- Claim 1, YOLOv4-tiny has significantly higher inference speed and slightly improved accuracy in comparison with other tiny models on MS COCO dataset.
- Claim 2, YOLOv4-CSP achieves better speed/accuracy trade-off compared with the original YOLOv4.
- Claim 3, YOLOv4-large has the highest accuracy on MS COCO dataset compared with any other algorithms.
- Claim 4, anchor box sizes of scaled-YOLOv4 algorithms are obtained through k-means and genetic algorithm, leading to better convergence effect and higher accuracy.
- Claim 5, CIoU loss is the best loss function with regard to accuracy for scaled-YOLOv4.
- Claim 6, Mish activation function has the overall best performance for scaled-YOLOv4.

Claim 3 is not considered in this paper as to test YOLOv4-large, the GPU performance can be very demanding.

III. Methodology

A. Implementation

The base code is provided by the authors in their paper. It was documented but not very in detail, and there exist some

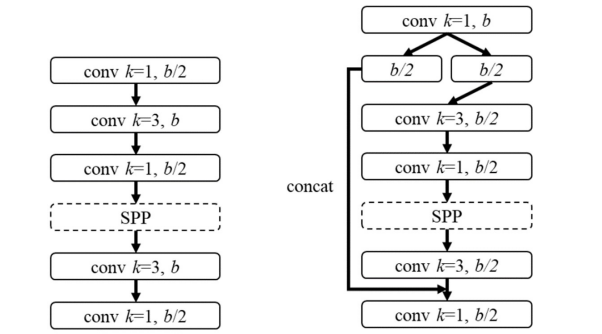
bugs that prevent the code from running successfully. After thoroughly reading and debugging the code, it can be used.

B. Model Descriptions

To investigate the aforementioned claims in the original paper, experiments are carried out in this study using scaled-YOLOv4. In this section, the general architectures of scaled-YOLOv4 algorithms tested are described.

Backbone network

Scaled-YOLOv4 generally adopts the concept of CSPNet in its backbone network to reduce the amount of computation, as shown in the computational block of CSPDarknet-53 used in YOLOv4-CSP in Fig. 1. The CSP-ization partitions the feature map of base layer into two parts and then merges them through a cross-stage hierarchy. Plenty of residual blocks exist in the backbone, and they are comprised of several convolution, batch normalization and activation function modules, which are reported to be capable of preventing vanishing gradient, exploding gradient and over-fitting problems.



(a) reversed dark layers (SPP) (b) reversed CSP dark layers (SPP)

Fig. 1. Computational blocks of original and CSP-ized layers

Activation function

Activation function is essential for neural network as it adds non-linearity and thus enhancing the expressive power. Several typical activation functions are listed below,

1) Sigmoid function,

$$\sigma(x) = 1/(1+e^{-x}) \tag{1}$$

2) Leaky Relu function,

$$LRelu(x) = \begin{cases} x, & \text{if } x \geq 0; \\ \alpha x, & \text{if } x \leq 0 \end{cases} \tag{2}$$

3) Swish function,

$$Swish(x) = x \cdot \sigma(\beta x) \tag{3}$$

4) Mish function,

$$Mish(x) = x \cdot \tanh(\ln(1+e^{x})) \tag{4}$$

As shown in Fig. 2, Mish is a differentiable function which is unbounded above and bounded below. Being unbounded above is desirable as it avoids saturation which causes training to drastically slow down due to near-zero gradients. Being bounded below is also advantageous as it results in strong regularization effects and reduces overfitting. Mish function is selected as the activation function in scaled-YOLOv4, and its true effect will be examined in this study.

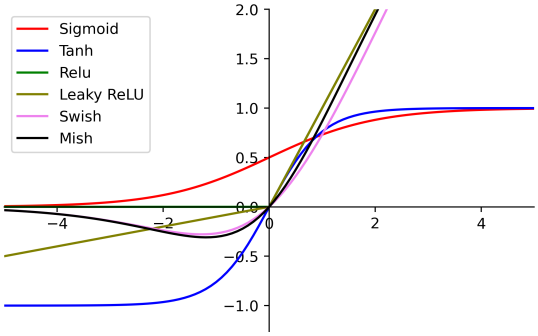


Fig. 2. Different typical activation functions

Loss function

After upsampling and feature fusion, the algorithm outputs three feature maps for detecting small-, medium- and large-size objects. For each feature map, each pixel has three anchor boxes with different shapes for bounding the targets. During training, the network outputs and trains each anchor box's center coordinate and size adjustment factors and object class. Commonly, prediction accuracy of bounding box is evaluated by intersection over union (IoU). IoU is defined by,

$$\text{IoU} = (A \cap B)/(A \cup B) \tag{5}$$

where A is the true bounding box and B is predicted bounding box, as shown in Fig. 3. Based on IoU, several loss functions are derived to evaluate training effect. GIOU loss is given by,

$$\begin{aligned} \text{GIoU Loss} &= 1 - \text{GIoU} \\ &= 1 - \text{IoU} + (C_w C_h - A \cap B)/C_w C_h \end{aligned} \tag{6}$$

DIOU is given by,

$$\begin{aligned} \text{DIoU Loss} &= 1 - \text{DIoU} \\ &= 1 - \text{IoU} + \rho(\boldsymbol{b}, \boldsymbol{b_{gt}})/C^2 \end{aligned} \tag{7}$$

where $\rho(\cdot,\cdot)$ is an operator for calculating Euclidean distance. CIOU is given by,

$$\begin{aligned} \text{CIoU Loss} &= 1 - \text{CIoU} \\ &= 1 - \text{IoU} + \rho(\boldsymbol{b}, \boldsymbol{b_{gt}})/C^2 + \gamma\mu \end{aligned} \tag{8}$$

where γ is positive trade-off parameter and μ is height-width consistency parameter. EIOU is given by,

$$\begin{aligned} \text{EIoU Loss} =& 1 - \text{EIoU} \\ =& 1 - \text{IoU} + \rho(\boldsymbol{b}, \boldsymbol{b_{gt}})/C^2 + \\ & \rho(w, w_{gt})/C_w^2 + \rho(h, h_{gt})/C_h^2 \end{aligned} \tag{9}$$

In the original paper it is claimed that CIoU has the best overall testing accuracy, and it is examined in this study. Other tricks in scaled-YOLOv4 algorithms such as mosaic image augmentation are also important, but they are not investigated in this study so they are not presented in detail here.

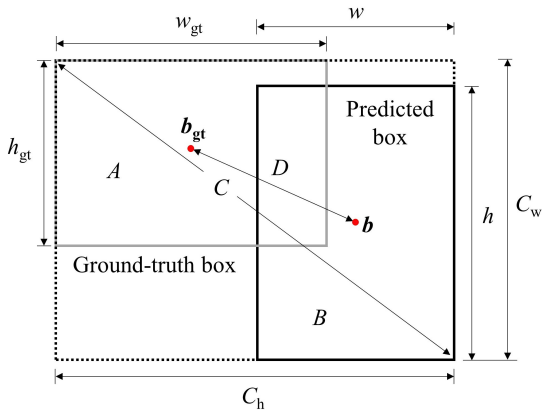


Fig. 3. Definitions of different IoU losses

C. Datasets

Two main datasets are used in the experiments in this paper: MS COCO 2017 and VOC 2007. In the original paper, all the results were obtained based on MS COCO 2017 dataset. However, due to insufficient computation power, MS COCO 2017 is too large for this reproducibility study, therefore part of the results are examined based on VOC 2007 dataset. In addition, by using another dataset, generalizability of conclusions in the original paper can be evaluated. Description of the datasets are listed below,

MSCOCO 2017: This is a large-scale image dataset containing 118K training images, 5k validation images and 41k test images and their corresponding labels of 80 classes of everyday objects and humans.

VOC 2007: This is a medium-scale image dataset containing 22k images and labels of 20 of classes of everyday objects and humans. In this study the training/validation sets are split by a ratio of 9:1.

D. Experimental Setup

A server with a single RTX 1080ti GPU and a server with a single Tesla V100 GPU are rented for reproducing claims 1 and 2. A single RTX 2080ti GPU that we own is also used for claim 1. Claims 4, 5 and 6 are tested on Tesla T4 and P100 GPUs on Google Colab Pro+.

IV. Results

In this section we describe the results of the experiments described in the methodology. We find that our experiments solidly support claims 1, 2 and 5, but we didn't find sufficient support for claims 4 and 6 across the datasets.

A. Claim 1

As shown in Table 1, 6 tests were conducted to examine the authenticity of claim 1, which involves two models: YOLOv4-tiny and YOLOv4-tiny (3l). In the original paper the definition of YOLOv4-tiny (3l) is not clear, and issue #25 in the authors' github explains YOLOv4-tiny (3l) is modified YOLOv4-tiny with three layers of network.

We tested the two models on a RTX 1080ti and a 2080ti GPU. The inference speed we obtained are all slightly slower than what is written in the original paper. Maybe it originates from the slight difference of hardware performance, as we did not test other models like YOLOv3-tiny for comparison. However, even though the inference speeds we get is slower, they are larger than the reported speeds of other tiny models, and this proves YOLOv4-tiny indeed has outstanding inference speed. The test results of precision values on MS COCO 2017 dataset are also slightly smaller than the corresponding reported values, but significantly larger than than that of other tiny models, thus claim 1 is proved authentic.

TABLE I
COMPARISON OF YOLOV4-TINY RESULTS

Model	Size	FPS_{1080ti}	FPS_{2080ti}	mAP@.5:.95
YOLOv4-tiny	640	196	252	24.2%
YOLOv4-tiny	416	356	421	21.5%
YOLOv4-tiny (3l)	320	244	312	28.3%
*YOLOv4-tiny	416	371	443	21.7%
*YOLOv4-tiny (3l)	320	252	-	28.7%
*ThunderS146	320	248	-	23.6%
*CSPPeleeRef	320	205	-	23.5%
*YOLOv3-tiny	416	368	-	16.6%

Notes: items with * are the results from original paper, FPS means frame per second, and mAP@.5:.95 means the average value of mean average precision under different IoU (from 0.5 to 0.95 with a stride of 0.05).

B. Claim 2

Two tests were conducted on a Tesla V100 GPU to examine claim 2. Surprisingly, all the precision values we get are larger than that is reported in the paper. It seems that the authors had reproducibility problem for issue #89 on their github, so they modified their codes afterwards. Compared with the original YOLOv4 algorithm, inference speed of YOLOv4-CSP is 1.5 times faster, and all of its precision values are improved. Therefore, the result of claim 2 is reproduced and proved authentic.

TABLE II
TEST RESULTS OF YOLOV4-CSP ON TESLA V100 GPU

Model	Size	FPS	AP	AP_{50}	AP_S	AP_L
YOLOv4-CSP	512	90	46.7%	65.1%	25.1%	61.8%
YOLOv4-CSP	640	68	48.6%	66.8%	33.0%	62.0%
*YOLOv4-CSP	512	97	46.2%	64.8%	24.6%	61.9%
*YOLOv4-CSP	640	73	47.5%	66.2%	28.2%	59.8%
*YOLOv4	320	62	45.5%	64.1%	27.0%	56.7%

Notes: items with * are the results from original paper, FPS means frame per second, AP means the average value of mean average precision under different IoU (from 0.5 to 0.95 with a stride of 0.05), AP_{50} means mean average precision under 0.5 IoU, AP_S means mean average precision for small objects, and AP_L means mean average precision for large objects.

C. Claim 4

In the original paper, the authors just said they used k-means combined with genetic algorithm to obtain the sizes of nine anchor boxes, but they didn't give any detail for this

process. We firstly used k-means method to get nine cluster centers of all the ground-truth box sizes in Table 3. Then, we made the genetic algorithm iterate around 1000 times, as shown in Fig. 5 and Table 3, the anchor size distribution becomes more similar with what is reported in the original paper, but no matter how we change the iteration number, we just cannot get exactly the same result.

By using the authors' anchor sizes on COCO dataset, the accuracy is indeed higher than that of our anchors. It is highly possible the authors used some other tricks for improvement, but in this study it is proved the results cannot be reproduced.

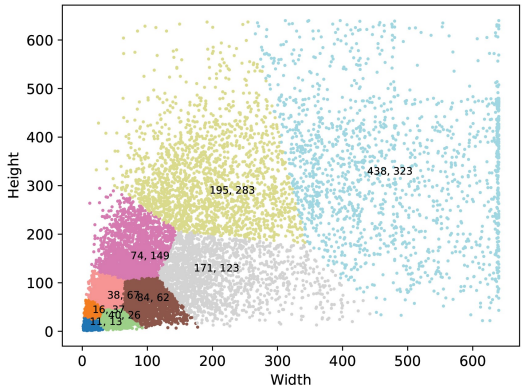


Fig. 4. Anchor clustering result using K-means and GA

TABLE III
COMPARISON OF ANCHORS FROM DIFFERENT CLUSTERING METHODS

Method	Anchor size	AP
K-means	(21, 22), (45, 61), (122, 68), (79, 144), (240, 150), (141, 266), (478, 206), (262, 410), (540, 442)	45.3%
K-means with GA for 1000 iterations	(11, 13), (16, 37), (40, 26), (38, 67), (84, 62), (74, 149), (171, 123), (195, 283), (438, 323)	47.9%
*K-means with GA	(12, 16), (19, 36), (40, 28), (36, 75), (76, 55), (72, 146), (142, 110), (192, 243), (459, 401)	48.6%

D. *Claim* 5

Due to the lack of computation power, the next two claims are examined on VOC dataset rather than MS COCO dataset. The results can still act as a proof of reproducibility as the authors claim Mish function and CIoU loss should be built in scaled-YOLOv4 for any dataset, so experiments on VOC dataset can test this claimed generalizability.

In the original paper, the authors compared the results of Mish and leaky Relu functions, but they didn't clarify the α value they used in leaky Relu, and they didn't compare with other popular activation functions.

As shown in Table 4, as the VOC dataset only has 20 classes, the precision results are significant higher that on the MS COCO dataset. It is shown that no matter the α value is 0.3 or 0.1, its accuracy is always smaller than that of Mish. Swith with a β value of 1 is the second best function with an AP of 62.3%, which is still 0.3% less than that of Mish. Thus, the authenticity of claim 5 is solidly proved.

TABLE IV
COMPARISON OF ACTIVATION FUNCTIONS ON YOLOV4-CSP

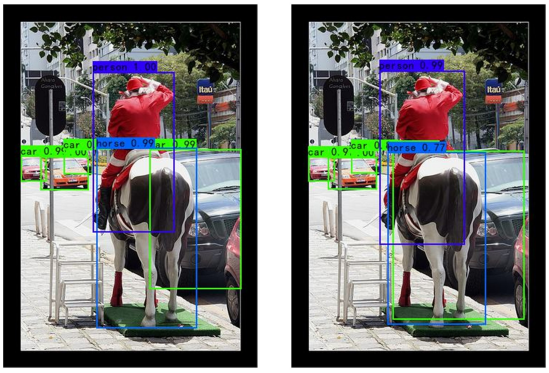
Backbone	Dataset	Size	Act. func.	AP
CSPDarknet-53	VOC 07	608	L.Relu 0.3	61.7%
CSPDarknet-53	VOC 07	608	L.Relu 0.1	61.3%
CSPDarknet-53	VOC 07	608	Mish	62.6%
CSPDarknet-53	VOC 07	608	Swish	62.3%
CSPDarknet-53	VOC 07	608	FRelu	62.1%
*CSPDarknet-53	MS COCO 17	608	L.Relu	46.9%
*CSPDarknet-53	MS COCO 17	608	Mish	47.5%

E. *Claim* 6

IoU losses are compared and EIoU loss has similar performance with CIoU loss with regard to AP_{50}, but it has even higher AP value. Thus, the authors' claim that CIoU has the best performance is not solid. In addition, the comparison between EIoU and GIoU results are shown in Fig. 5, and EIoU generally has better bounding box location and size.

TABLE V
COMPARISON OF LOSS FUNCTIONS ON YOLOV4-CSP

Loss	AP_{50}	AP
GIOU	78.8%	60.6%
DIOU	81.9%	63.2%
CIOU	82.6%	63.7%
EIOU	82.5%	64.3%



(a) EIoU loss (b) GIoU loss

Fig. 5. Comparison of bounding boxes using different loss functions

V. CONCLUSIONS

It can be concluded that the main claims in the scaled-YOLOv4 paper are authentic, as YOLOv4-tiny indeed has very high inference speed and YOLOv4-CSP has improved accuracy and speed compared with its predecessor, YOLOv4. Unfortunately, with our test setup it is too expensive to test the effect of YOLOv4-large, but this is something worth testing in the future. With regard to the details of the algorithms, there are some slight deviations that cannot be reproduced, and it is highly possible that the authors incorporated some tricks they did not mention in the paper. Finally, it is found that the algorithm can still be improved by EIoU loss.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[2] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[3] ——, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[5] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in Proceedings of the IEEE/cvf conference on computer vision and pattern recognition, 2021, pp. 13 029–13 038.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision. Springer, 2016, pp. 21–37.

[7] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," arXiv preprint arXiv:1701.06659, 2017.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[9] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Advances in neural information processing systems, vol. 28, 2015.

[11] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.