# On the Importance of Large Objects in CNN Based Object Detection Algorithms

Ahmed Ben Saad[1,2] Gabriele Facciolo[1] Axel Davy[1]

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, 91190, Gif-sur-Yvette, France
[2]Schlumberger AI Lab

## Abstract

*Object detection models, a prominent class of machine learning algorithms, aim to identify and precisely locate objects in images or videos. However, this task might yield uneven performances sometimes caused by the objects sizes and the quality of the images and labels used for training. In this paper, we highlight the importance of large objects in learning features that are critical for all sizes. Given these findings, we propose to introduce a weighting term into the training loss. This term is a function of the object area size. We show that giving more weight to large objects leads to improved detection scores across all object sizes and so an overall improvement in Object Detectors performances (+2 p.p. of mAP on small objects, +2 p.p. on medium and +4 p.p. on large on COCO val 2017 with InternImage-T). Additional experiments and ablation studies with different models and on a different dataset further confirm the robustness of our findings.*

## 1. Introduction

Object detection is a fundamental task in computer vision with applications in a variety of fields (autonomous vehicles, surveillance, robotics, ...). It has been studied in computer vision since the dawn of automatic Image Processing [7, 15, 41]. The surge of Convolutional Neural Networks (CNNs) [19] has revolutionized the field leading to a proliferation of methods [10, 21, 31, 38, 45] and substantial improvements in detection scores.

Researchers have proposed several variants of object detection models, including one-stage [22, 23, 31, 39] and two-stage detectors [9, 10, 21, 29], to improve the speed and accuracy of object detection. Furthermore, novel techniques, such as attention mechanisms [4, 24, 40] and anchor-free object detection [5, 18, 39], have emerged to further improve the performances of existing models. In this paper, we aim to focus on object detection models and analyze their underlying mechanisms for locating objects within an image.

Detection datasets usually contain a large number of easy examples and a small number of hard ones. Automatic selection of these hard examples can make training more effective and efficient [36]. Different data sampling techniques were proposed depending on the criterion for selecting the hard samples during training. These criteria include high current training loss [37], Foreground/Background ratio unbalance [22, 34], IoU-unbalance shifting towards hard examples [27] and class unbalance [28].

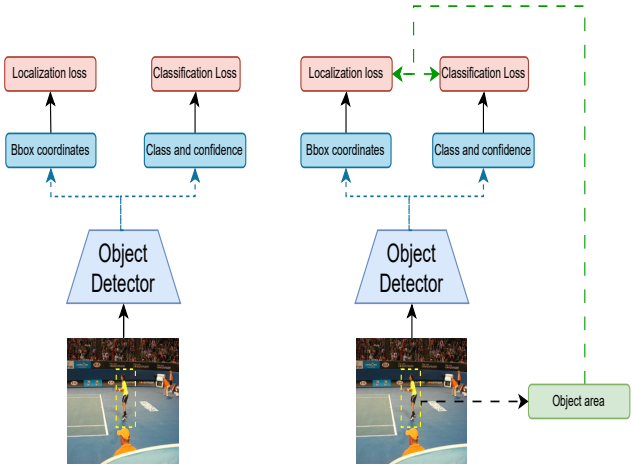


Figure 1. Overview of the proposed weighting policy (right) compared to a general object detection framework (left). The area of each bounding boxes is computed (black dotted arrow) then its log is taken as a sample weight for the corresponding bounding box in both classification and localization losses (green dotted arrows). This gives more importance to large objects and performances across all sizes benefit from it.

The influence on detection performance of object size distribution of a training dataset is less examined subject in the literature. Common wisdom would dictate that if the final goal is to have a maximum performance for a given size of objects - say small objects - more emphasis during training should be given to these target objects. Our work shows that reality can be counter-intuitive, as we find that giving more focus to large objects can improve performance for all

object sizes, including small ones. Indeed, we find that a simple change in the training loss can increase performance for various object detectors. The loss functions of object detection can be categorized as two sorts: the classification loss and the localization loss. The first is used to train a classification head that will detect and, in the case of multiclass object detection, categorize the target object. The second is used to train a head that will regress a rectangular box to find the target object. We propose to incorporate the sample weight function in the total loss computation, including the classification term (see Figure 1). By assigning less weight to smaller objects and more weight to larger objects, the model learns effectively from both small and large objects.

Through empirical evaluations and ablations, we validate the effectiveness of the proposed weight function and demonstrate its potential for advancing the state-of-the-art in object detection. Our contribution are the following:

- We verify that learning on large objects leads to better detection performance than learning on small objects.
- We propose a simple loss re-weighting scheme that gives more emphasis to large objects, which results in an overall improvement of object detectors' performances across all objects sizes.
- We analyze for which object detection sub-tasks the performance gains are most seen, improving the understanding of the impact of the loss re-weighting.

## 2. Related work

Besides the use of geometrical data augmentation techniques, over the years object detector architectures have incorporated more and more elements to improve performance across object scales. In this section, we review some of the models that we deem important for their influence or performances. Mainly highlighting the proposed ideas to deal with the different object sizes. We then focus on data augmentation, how it has been used for the same goal and its limitations.

**Feature Pyramid Networks (FPN)** Feature Pyramid Networks (FPN) is a widely used module proposed by Lin et al. [21] that addresses the limitations due to having one common prediction output for all object scales. More specifically it proposed to extract features at different levels of a backbone convolutional network [21, 30, 38], and merge them back in an inverted feature pyramid. Then each level of the inverted feature pyramid has a dedicated detection branch dedicated to objects of a given size range. The performance gains can be attributed to capturing semantic information at higher resolutions while maintaining spatial information at lower resolutions.

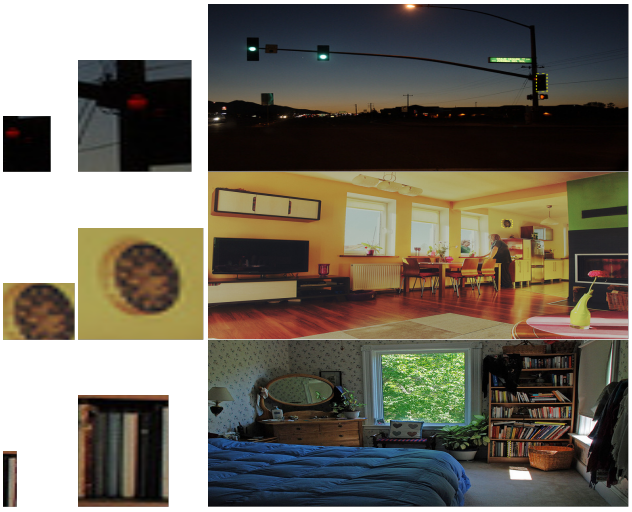

Figure 2. Example of some small objects cropped without or with small context (first and second columns) and their entire context in the image (third column, the objects are highlighted in yellow bounding boxes, zoom for better view). We focus on a traffic light in the first row, a clock in the second and a book in the third. We see from these examples that in the case of small objects, it is difficult, even to a human eye, to correctly label the designated object. Also, the more context we have, the easier is the classification. This also applies to CNNs.

**YOLO** YOLO (You Only Look Once), proposed by Redmon et al. [31], is a real-time anchor based one-stage object detection system that uses a single neural network to simultaneously predict object bounding boxes and class probabilities in real time and directly from input images. Achieving state-of-the-art speed and accuracy. Since its inception, YOLO has undergone several evolutions to enhance its performance. YOLOv2 [32] improved upon the original architecture by introducing anchor boxes to enable the model to efficiently detect objects of different aspect ratios and sizes. YOLOv3 [33] incorporated a feature pyramid network, which enabled the model to effectively capture objects at multiple scales. YOLOv4 [2] adopted the CSPDarknet53 [43] backbone, which improved the model's capacity to extract complex features. It also incorporated the PANet [44] module, which performed feature aggregation across different levels of the network,further improving object detection at various scales. YOLOv5 [13] is a PyTorch implementation of YOLO, characterized by practical quality-of-life improvements for training and inference. In terms of performance it is comparable to YOLOv4.

**TTFNet** TTFNet [25] is a derivative of CenterNet [47], which defines an object as a single point (the center point of its bounding box). It uses keypoint estimation to find center points and regresses to all other object properties. TTFNet speeds-up the training of CenterNet by predicting bound-

ing boxes not only at the center pixel but also around it using a Gaussian penalization. Several weighting schemes were considered and the authors found that the best performance was reached by normalizing the weights then multiplying by the logarithm of the area of the box. The localization loss is then normalized by the sum of the weights present in the batch. Inspired by this approach we propose to add the logarithmic weighting also to the other terms, namely localization and classification. Other works such as FCOS [39], have studied the impact of the bounding boxes areas on the training, but to the best of our knowledge, none have proposed a weighting scheme to focus on large objects. In FCOS all the pixels of the bounding box contribute to its prediction, but the subsequent loss is averaged among all pixels. Its implementation was later extended as FCOS Plus,[1] which reduces the learning region to a center region inside the box.

**DETR** DETR (Detection Transformer) [4] introduces a transformer-based architecture to object detection that enables simultaneous prediction of object classes and their bounding box coordinates in a single pass. Notably, DETR utilizes a global loss function based on sets, allowing it to effectively handle variable object counts through the integration of self-attention mechanisms and positional encodings.

**InternImage** InternImage, proposed by Wang et al. [45], is a large-scale CNN-based foundation model which capitalizes on increasing the number of parameters and training data, similar to Vision Transformers [8]. InternImage employs deformable convolutions [6] as its core operator, allowing it to capture richer contexts in object representations. Moreover, InternImage incorporates adaptive spatial aggregation conditioned by input and task information, reducing the strict inductive bias commonly observed in traditional CNNs. InternImage has attained improved object detection results and currently holds high ranks in evaluation scores across different datasets. As we will see we can further improve the performance of InternImage by training with a size-dependent weighting term.

**Data Augmentation** Data augmentation is a powerful solution to enhance the performance of object detection models across all object sizes [14, 35]. By applying transformations to the training dataset, data augmentation techniques introduce diversity and expand the representation of objects at different scales. Augmentations such as random scaling, flipping, rotation, and translation enable the models to learn robust features for accurate detection of both small and large objects. Augmentations specifically designed for small objects, such as random patch copy-pasting and pixel-level augmentations [17], help alleviate issues related to low-resolution details and limited contextual information. Similarly, augmentations that preserve spatial context and prevent information loss during resizing or cropping [26,46] assist in handling large objects. However, it is important to note that data augmentation techniques have limitations when it comes to object sizes. While augmentations can introduce diversity and expand the representation of objects, upscaling objects does not inherently yield additional information. Increasing the size of small objects through augmentation may improve their visibility, but it does not provide additional contextual details or features that were not present in the original image. On the other hand, downscaling or resizing larger objects can potentially lead to the loss of important information and fine-grained details, which may hinder accurate detection.

[1] https://github.com/yqyao/FCOS_PLUS

| | Range | ratio$_{train}$ | ratio$_{val}$ |
|---|---|---|---|
| Small | $[0, 32^2]$ | 0.27 | 0.28 |
| Medium | $[32^2, 96^2]$ | 0.45 | 0.44 |
| Large | $[96^2, +\infty[$ | 0.27 | 0.29 |

Table 1. Objects sizes ranges in COCO dataset.

Little attention has been given to the content of the dataset itself (with exception of annotation errors [1, 11]), in particular the impact of object size distribution on the detection performance across all scales. In the next section, we highlight the importance of features learned from large objects on the overall performances of object detectors.

## 3. On the importance of objects sizes

Datasets such as COCO incorporate a diverse set of objects of various sizes. However, detecting large objects presents different challenges compared to small ones. Large objects have rich details and texture, which might have to be interpreted or ignored, but this rich information are usually enough to know what they are without surrounding context. Small objects differ in that the surrounding context has significant importance in their interpretation. As an illustration of this fact, Figure 2 shows a set of cropped small objects without or with their context. We tend to imagine that small object detection depends mainly on the earlier stages of a backbone. However, this observation implies that the latest stages of the backbone have features that capture large objects, but also the context needed to detect small ones. As a result all object sizes need good quality features at all levels of the network backbone. The intuition behind our research is that having a variety of object sizes helps learn high quality features at all sizes, and that emphasizing the importance of large objects in the loss is even better.

This intuition can be verified by the following experiment: Given an object detector (YOLO v5 [12] in this case)

| Scores | Small | | Medium | | Large | | All | |
|---|---|---|---|---|---|---|---|---|
| | mAP | mAR | mAP | mAR | mAP | mAR | mAP | mAR |
| Only pretrain on Small+Medium | 0.265 | 0.425 | 0.451 | 0.676 | 0.401 | 0.628 | 0.281 | 0.512 |
| Only pretrain on Large | 0.187 | 0.356 | 0.325 | 0.605 | 0.482 | 0.731 | 0.255 | 0.426 |
| Pretrain on Small+Medium then finetune on all | 0.253 | 0.413 | 0.424 | 0.667 | 0.446 | 0.695 | 0.296 | 0.521 |
| Pretrain on Large then finetune on all | 0.271 | 0.433 | 0.487 | 0.681 | 0.542 | 0.753 | 0.350 | 0.604 |
| Reference scores (Train directly on the entire dataset) | **0.297** | **0.456** | **0.540** | **0.718** | **0.674** | **0.833** | **0.372** | **0.660** |

Table 2. Test mAP and mAR scores for different pretraining object sizes on COCO val 2017. The finetuning step (if it exists) is done while freezing the encoder part of the network. Note that pretraining on large objects improves the scores for all sizes compared to pretraining on small/medium objects

and a training dataset (COCO [20]) we start by initializing the model with random weights and pretrain it using only large objects. We used the size ranges used by the authors of YOLO v5 in their github repository[2] and shown in Table 1. We then freeze the encoder layers and fine-tune the model on all the training data. We also repeat the same procedure but using the small and medium data for pretraining. The results of train and test mAP and mAR are shown in Table 2. The goal of these experiments is to observe the quality of the learned backbone features for various object sizes when trained exclusively on large or small+medium objects.

We can see that, despite the relatively lower quantity of large objects compared to the rest of the dataset, the model pretrained on large objects and finetuned on the entire dataset performs better across all sizes. This means that features for bigger objects are more generic and can be used to detect at all object sizes including smaller ones. This is less the case for features learned on small objects.

Another interesting point is that the network trained only on small and medium objects performs worse on these objects than the network trained on the whole dataset. In fact even the network using the backbone pretrained only on large objects and finetuned on the entire dataset has better detection performance on small objects. This highlights the argument that large objects help learn more meaningful features for all scales.

# 4. Proposed method

## 4.1. Weight term

To effectively leverage the large-sized objects for enhancing model performance, we propose the inclusion of a weight term in the loss functions specifically designed for object detection tasks

$$W_i = log(h_i \times w_i), \tag{1}$$

where $h_i$ is the i-th object height and $w_i$ is its width.

For example, let us consider the YOLO v5 loss

$$L_{total} = \lambda_1 L_{classif} + \underbrace{L_{confidence} + \lambda_2 L_{CIoU}}_{L_{detection}}. \tag{2}$$

[2] https://github.com/ultralytics/yolov5

At each training step, the loss is calculated as an average over all batch samples

$$L_{a,batch} = \frac{1}{N_b} \sum_{i \in B_{batch}} L_\psi(i, \hat{i}) \tag{3}$$

with $\psi \in \{confidence, classif, CIoU\}$, $N_b$ the number of bounding boxes in the batch, $B_{batch}$ the set of the bounding boxes in a batch, $i$ the prediction over one bounding box and $\hat{i}$ the corresponding ground truth. We modify $L_{\psi,batch}$ to incorporate the weights $W_i$ with

$$L^{new}_{\psi,batch} = \sum_{i \in B_{batch}} \alpha_i L_\psi(i, \hat{i}), \tag{4}$$

where $\alpha_i = \frac{W_i}{\sum_{k=1}^{N_b} W_k}$. This term aims to assign higher weights to larger objects during the training and thus encourages the models to focus more on learning from them. On the other side, small objects get a reduced impact on learning, as the sum of the weights in the batch is normalized. Yet, the slow increase of the logarithm means that no object size is negligible in the loss.

As mentioned in Section 2, the weighting term (4) was already used in TTFNet. However, contrary to TTFNet, which incorporated this weight in its size regression loss (GIoU), we use it on both the localization and classification loss terms. We justify this choice by an ablation study in Section 6.1.

The inclusion of the weight term in the loss functions encourages the models to prioritize the accurate detection and localization of large objects. This leads to more discriminative features and better contextual understanding, particularly concerning larger objects. And as a consequence, the models become better equipped to handle also small objects.

Furthermore, the weight term helps to address the inherent dataset bias towards smaller objects by explicitly giving larger objects more prominence during training. This bias correction allows the models to learn more effectively from the limited number of large objects present in the dataset, bridging the performance gap between small and large object recognition. For example, in Table 3, the ratio of each

| Dataset | Small | | Medium | | Large | |
|---|---|---|---|---|---|---|
| | $r$ | $r'$ | $r$ | $r'$ | $r$ | $r'$ |
| COCO | 0.28 | 0.13 | 0.44 | 0.33 | 0.29 | 0.54 |
| NuScenes | 0.39 | 0.15 | 0.46 | 0.38 | 0.15 | 0.47 |

Table 3. Comparison of $r_{size}$ and $r'_{size}$ across all sizes on COCO and NuScenes, note how the weighted ratios $r'$ are shifted towards large objects compared to the normal ratios of objects.

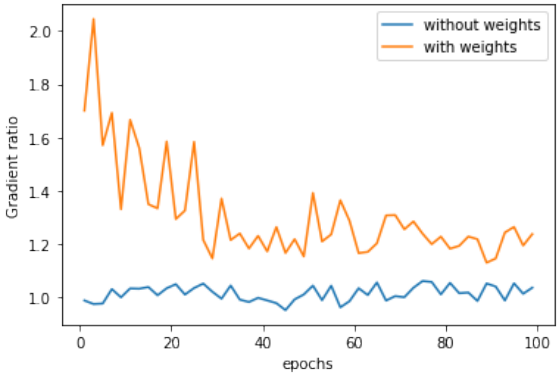


Figure 3. Evolution of the ratio of the sum of gradient amplitudes $r_{grad}(\theta)$ over the first 100 epochs on COCO for YOLO v5. We see that the weighting term makes the impact of large object greater than small objects, resulting in an overall increase in performances.

object size

$$r_{size} = \frac{\#Objects_{size}}{\#Objects} \tag{5}$$

is compared to the weighted sum of these objects

$$r'_{size} = \frac{\sum_{object \in size} log(w_{object} h_{object})}{\sum_{Objects} log(w_{object} h_{object})} \tag{6}$$

on COCO and NuScenes [3] datasets. We see that $r'$ is shifted towards large objects despite the actual ratio of those objects being relatively small. This forces the training to focus more on large objects, which benefits performance across all sizes. This raises the question of the ideal ratios for the distribution of object sizes when building a dataset, and likely this will depend on the target objects and their complexity at different sizes. Thus each dataset might have a different optimal weighting function.

### 4.2. The effect of the weight term on the training

In order to gain more insight on the effect of the weighting term on the training, we need to quantify the importance of each sample during training. The authors of [42] argue that the sum of the gradient magnitude of the loss can be a good measure of this. In fact, the evolution of the parameters of the model $\theta$ during training is proportional to the magnitude of gradient of the loss w.r.t the model parameters $\left\|\sum_{i \in Batch} \nabla_\theta L_{i,\theta}\right\|$. Since these gradients live in a high

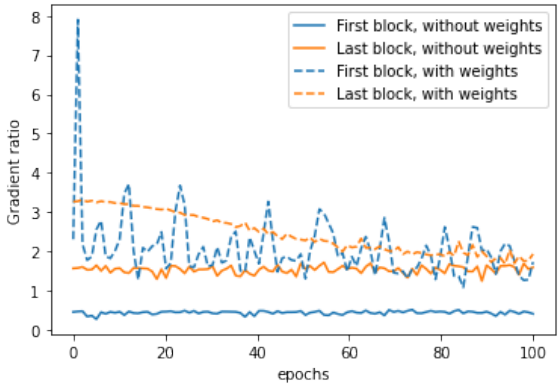


Figure 4. Evolution of $r_{grad}(\theta_{block})$ restricted to the first and last BottleNeckCSP blocks for YOLO v5 over the first 100 epochs on COCO. We see that the both layers (especially the first one) are affected by the weighting function.

dimensional space, any two gradient vectors associated to two inputs are likely orthogonal. Therefore the triangular inequality

$$\left\| \sum_{i \in Batch} \nabla_\theta L_{i,\theta} \right\| \leq \sum_{i \in Batch} \|\nabla_\theta L_{i,\theta}\| \tag{7}$$

can be used as a tight estimate of the weights update. Thus, we can consider $\|\nabla_\theta L_{i,\theta}\|$ as a measure of the impact of each object on the learned features and we can regroup these quantities by object size to see the effect of each object size on the learning procedure. We computed the ratio of the sum of gradient magnitudes of large objects over those of small objects

$$r_{grad}(\theta) = \frac{\sum_{i \in \Omega_{Large}} \|\nabla_\theta L_{i,\theta}\|}{\sum_{i \in \Omega_{Small}} \|\nabla_\theta L_{i,\theta}\|}, \tag{8}$$

where $\Omega_{Large}$ is the set of large objects, $\Omega_{Small}$ is the set of small objects and $L_{i,\theta}$ is the training loss term evaluated and the input $i$ (before the reduction over the image and the entire batch).

Figure 3 shows the evolution of this ratio along 100 epochs for YOLO v5 on COCO, using or not using the proposed weighting term. We can see that, without the weighting term, small and large object have an comparable contribution on the model parameters. This translates to $r_{grad}(\theta)$ oscillating around 1. In contrast, using the weighting increases the impact of larger objects. This is shown by the value of $r_{grad}(\theta)$ starting high (at about 1.8) at the beginning of the training and stays larger than 1 as the training continues.

To further investigate this effect, we studied this behavior at different levels of the network. The YOLO v5 architecture is based on 7 BottleNeckCSP blocks: two of them form the backbone and the others are the main component of the

model's neck (the PANet part). We restrict the analysis to the parameters of the first or last BottleNeckCSP blocks and define

$$r_{grad}(\theta_{block}) = \frac{\sum_{i \in \Omega_{Large}} \|\nabla_{\theta_{block}} L_{i,\theta_{block}}\|}{\sum_{i \in \Omega_{Small}} \|\nabla_{\theta_{block}} L_{i,\theta_{block}}\|}, \quad (9)$$

where $\theta_{block}$ is the set of parameters of a given BottleNeckCSP block of the model. Figure 4 shows the evolution of $r_{grad}(\theta_{block})$ for the parameters of the first or last BottleNeckCSP blocks.

This provides insights about the effects on low-level and high-level features. We see that the first block is particularly impacted when the weighting function is used, with the ratio increasing up to 16 folds at the beginning of the training and stabilizing at a 4-fold increase later on. For the last layer, we still observe an increase of $r_{grad}$, but less important. This suggests that focusing the training on large objects impacts mostly the low-level features and does so during all the training. One can argue that these generic low-level features are more distinguishable on large objects than on small ones.

These findings shed some light on how the re-weighting affects the training, suggesting that low level features are benefiting the most from large objects. In addition, one can argue that the shift of focus towards large objects is related to the overall performance improvement as this is observed since the first training epochs (this will be discussed in the next section).

## 5. Experiments

To corroborate the impact of the proposed weighting scheme we compare the performance of several object detectors: YOLO V5, InternImage, DETR [4] and Mask R-CNN [10] on the COCO and nuScenes datasets, with and without the weight term. We trained these models on both datasets on two NVIDIA RTX 2080 Ti for 35 epochs each with a batch size of 16. We used a warm-up of 5 epochs for InternImage-T. We used the Adam optimizer [16] with a Cosine Annealing learning rate starting from a max value of 0.01 for YOLO v5 and Mask R-CNN and 0.1 for InternImage-T and DETR. The minimum IoU for validating a detection is fixed to 0.5 and a confidence threshold of 0.001 for COCO and 0.05 for nuScenes. As for data augmentation, we kept the same pipeline defined for each method on their respective papers.

The results of these experiments, in terms of mAP and mAR scores are shown in Table 4. We see that all models exhibit a significant performance improvement across all object sizes when using the proposed weighting scheme. For instance, InternImage-T with the proposed changes reaches 51.2% mAP, while the original had 47.2% mAP, which is a 4 p.p. gain. Our base results reproduce the results of InternImage's authors, and their paper shows that

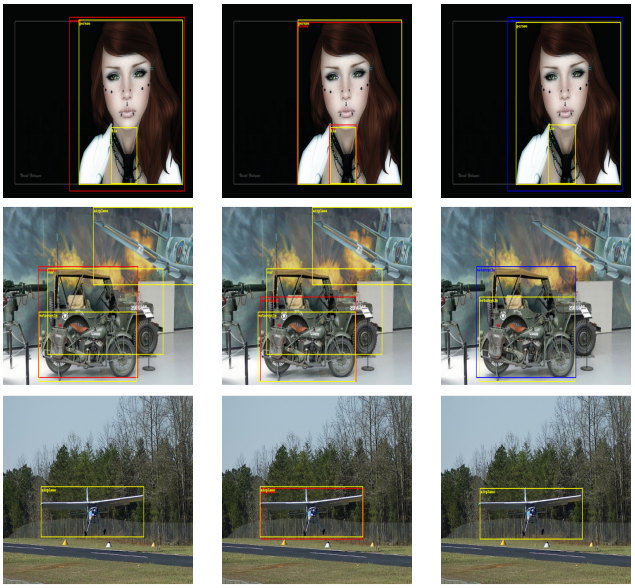

Figure 5. Qualitative display of some of the improvement made by adding the sample weight term. Columns from left to right: Without weights (Red: Prediction, Yellow: GT), With weights (Same Colors), Comparison (Blue: Without weights, Yellow: With weights)

InternImage-B, which has more than double the number of parameters than InternImage-T, only reaches 48.8% under similar training. We couldn't train with our modifications their biggest model InternImage-XL, which is the state of the art at the time of writing, as it requires expensive training resources. It is likely training such a model would define a new state of the art. While the results are shown here on four different CNN object detectors, the proposed weighing scheme is quite simple and can be applied easily to other object detection models.

A qualitative comparison is also shown in Figure 5. The selected examples show that the proposed modification allows the model to detect some objects that were otherwise undetected. For example, on the first and third rows, a tie and a plane are detected, respectively, only on the model with our modification. Bounding box predictions are also improved, as can be seen for example on the first and second row, where objects detected by both models have a more precise bounding box on the second column.

We also validate the improvement on another dataset: NuScenes. We used InternImage as model and compared its performance with and without the weight term. The results are shown in Table 5. We observe that we still have a slight improvement in the scores with the weighted loss. The evolution of the overall mAP w.r.t the number of epochs, shown in Figure 6, proves also that the model benefits from focusing on big objects since the start of the training as the performance is consistently better along the training proce-

| | | | Small | | Medium | | Large | | All | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | #Params | Weighted loss | mAP | mAR | mAP | mAR | mAP | mAR | mAP | mAR |
| Mask R-CNN ResNet-50 FPN | 44M | No | 0.223 | 0.386 | 0.485 | 0.621 | 0.513 | 0.697 | 0.364 | 0.512 |
| | | Yes | **0.248** | **0.403** | **0.518** | **0.641** | **0.560** | **0.724** | **0.393** | **0.545** |
| YOLO v5 | 64.4M | No | 0.297 | 0.456 | 0.520 | 0.640 | 0.617 | 0.745 | 0.372 | 0.599 |
| | | Yes | **0.315** | **0.462** | **0.549** | **0.702** | **0.654** | **0.781** | **0.398** | **0.623** |
| DETR ResNet-50 | 41M | No | 0.312 | 0.459 | **0.531** | 0.675 | 0.628 | 0.776 | 0.420 | 0.553 |
| | | Yes | **0.331** | **0.460** | 0.529 | **0.697** | **0.659** | **0.802** | **0.440** | **0.570** |
| InternImage-T | 49M | No | 0.309 | 0.464 | 0.538 | 0.718 | 0.674 | 0.833 | 0.472 | 0.66 |
| | | Yes | **0.332** | **0.487** | **0.556** | **0.736** | **0.714** | **0.857** | **0.512** | **0.679** |

Table 4. Models performances on COCO val 2017. We can see from the results that the introduction of the sampling weight term improved the models scores across all sizes.

| | | | Small | | Medium | | Large | | All | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | #Params | Weighted loss | mAP | mAR | mAP | mAR | mAP | mAR | mAP | mAR |
| InternImage-T | 49M | No | 0.551 | 0.634 | **0.573** | **0.710** | 0.652 | 0.738 | 0.648 | 0.711 |
| | | Yes | **0.562** | **0.661** | 0.570 | 0.708 | **0.679** | **0.762** | **0.659** | **0.724** |

Table 5. InternImage-T performances on NuScenes. These results show that the benefits of the weighting term are reproducible on different datasets. The amount of progress may depend on the ratios of small and large objects.

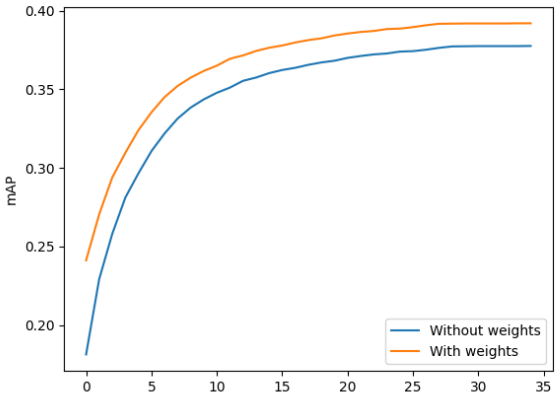


Figure 6. The addition of a weighting term to the detection and localization loss improves the validation scores on COCO val 2017 since the beginning of the training.

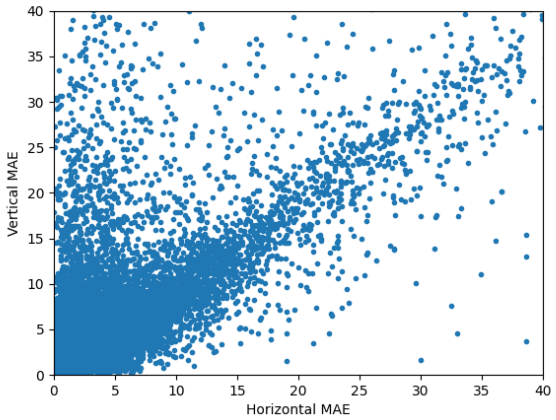


Figure 7. The horizontal and vertical MAEs are highly correlated for YOLO v5 on COCO, with a correlation coefficient of 0.7710

dure. We can see that from the first epochs, our weighting policy yields an improvement of nearly 3 p.p. on average. This emphasizes the intuition that the increased presence of large objects helps steer training in a better direction and avoids worse local minima. This also indicates that the effect of future improvements to object weighting could be seen early in training.

## 6. Ablation studies and discussion

### 6.1. Impact on the terms of the loss

To further investigate the impact of weighting strategies in the YOLO v5 loss function, we conducted an ablation study on the COCO dataset. Given the total loss function of the model (2), we vary the use of the weighting function for both $L_{classif}$ and $L_{detection}$. More specifically, we explored four scenarios: no weight terms, weight terms applied to the classification term only, weight terms applied to the detection term only and the weight term applied to all the loss terms.

Our analysis focuses on evaluating the Mean Average Precision (MAP@50:95) as a general metric score and the error on the bounding box center as a localization metric score. Table 6 shows the impact of each combination on the mAP for various sizes of objects. As the mAP is impacted by both the localization errors and the capacity of the network to detect the objects and correctly classify them, we complement the results the Mean Absolute Error (MAE: average L1 distance of the predicted bounding box center compared to the ground truth center). The MAE was estimated only on the horizontal component. We can justify this by the high correlation between horizontal and vertical MAEs (see Figure 7). In order to reduce the impact of the

| $L_{classif}$ | $L_{detection}$ | $mAP_{small}$ | $mAP_{medium}$ | $mAP_{large}$ | $mAP_{all}$ | $MAE_{small}$ | $MAE_{medium}$ | $MAE_{large}$ | $MAE_{all}$ | $AP@50_{all}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| - | - | 0.297 | 0.540 | 0.674 | 0.372 | 9.843 | 12.427 | 14.824 | 12.523 | 0.572 |
| ✓ | - | 0.290 | 0.542 | 0.680 | 0.371 | 8.245 | 11.346 | 11.293 | 10.576 | 0.584 |
| - | ✓ | 0.315 | 0.551 | 0.686 | 0.384 | 0.315 | 0.551 | 0.686 | 0.384 | 0.587 |
| ✓ | ✓ | **0.332** | **0.556** | **0.714** | **0.398** | **5.645** | **9.587** | **8.386** | **8.231** | **0.615** |

Table 6. Ablation study on the introduction of the weight term in classification and detection loss term and its effect on the mAP@50:95, the Mean Absolute Error (Pixels) on the bounding box center and the Average Precision on IoU=0.5 on COCO val 2017 dataset.

| Sample weight function | mAP | mAR |
|---|---|---|
| $f(w \times h) = 1$ | 0.372 | 0.599 |
| $f(w \times h) = w \times h$ | 0.217 | 0.412 |
| $f(w \times h) = \sqrt{w \times h}$ | 0.243 | 0.459 |
| $f(w \times h) = log(w \times h)$ | **0.398** | **0.623** |

Table 7. Results of different sample weights functions on COCO 2017 val using YOLO v5.

capacity of the network to detect objects, these results were computed on the set of objects correctly detected (correct class and IOU $> 0.5$). Lastly as AP@50 is less sensitive to localization errors, we display the corresponding results across all objects.

The results show that when adding only the weighting scheme to the classification term, the mAP regresses slightly, in particular for small objects, despite improved AP50 and MAE. The exact interpretation of this phenomenon is unclear. However, when the changed term is the detection term, mAP, MAE and AP50 are improved. The MAE gain is more important relatively for large objects (30%), indicating better localization. Lastly, having the weighting scheme on both loss terms gives the best performance on all metrics. Proportionally compared to the initial results, the highest gain is seen on small targets, as it sees a 12 p.p. increase in mAP (against 3 p.p. for medium and 6 p.p. for large objects) and a 43% decrease in MAE (against 23% for medium and 36% for large objects).

This suggests that a holistic approach that considers both classification and detection, with the weight terms appropriately assigned, is crucial for achieving the best results in terms of mAP score and bounding box center error.

### 6.2. On the choice of $log(w \times h)$

As discussed above, the main idea behind the choice of $log(w \times h)$ is to increase the contribution of large objects in the learning of the network features. We tested other functions of $w \times h$ and compared them to the proposed function. Table 7 evaluates some sample weighting functions for YOLO v5 on the COCO dataset. We kept on the idea that this function should depend on the areas of the objects and changed only the type of function (linear, logarithmic, square root). Although $log(w \times h)$ yields the best results in this table, we believe that additional research and experimentation is required in this direction in order to identify better functions or to prove that the chosen weight function is the optimal choice for better performances.

### 6.3. Impact of the dataset

The performance gain was demonstrated on two datasets: COCO and NuScenes. While the performance gain on these two datasets is far from negligible, there is no guarantee that similar gains can be obtained on other datasets. In fact the weighting scheme comes down to artificially increase the proportion of larger objects in the dataset, and thus if the dataset already had optimal proportions, the weighting wouldn't increase the performance. The conclusions from this research though is that when building a dataset, it is important to have a significant proportion of large objects, and if not, compensate with a weighting factor. One aspect impacting weighting needs is the difficulty of the detection of each object's size. For COCO and NuScenes, the detection scores for small objects are lower than for large objects. As small objects are harder to detect they tend to have stronger errors in the loss, and thus higher gradients. The weighting scheme can be seen as a correction factor to this behavior.

## 7. Conclusion

In this paper, we have shown that the presence of large objects in the training dataset helps to learn features that yield better performance also on small and medium objects. We then proposed a simple loss reweighting scheme that leads to improved performance of object detectors. Our findings underscore the importance of considering large objects and demonstrate the potential of incorporating a weighted loss term in enhancing overall object detection performance. Through experiments and ablation studies, we validated the effectiveness of our proposed approach. We evaluated different models and datasets, consistently observing improvements in detection scores across all sizes.

Future research in this area could investigate novel strategies that explicitly consider the impact of large objects on detection accuracy across different scales.

**Acknowledgments** We acknowledge support from ANRT CIFRE Ph.D. scholarship n°2020/0153 of the MESRI. This work was performed using HPC resources from GENCI–IDRIS (grant 2023-AD011011801R3) and from the "Mésocentre" computing center of CentraleSupélec and ENS Paris-Saclay supported by CNRS and Région Île-de-France (http://mesocentre.centralesupelec.fr/).

## References


[1] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11700–11709, 2019. 3

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. 2

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 5

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 1, 3, 6

[5] Gong Cheng, Jiabao Wang, Ke Li, Xingxing Xie, Chunbo Lang, Yanqing Yao, and Junwei Han. Anchor-free oriented proposal generator for object detection. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–11, 2022. 1

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *CoRR*, abs/1703.06211, 2017. 3

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 6

[11] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012. 3

[12] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, Yonghye Kwon, Kalen Michael, Jiacong Fang, Zeng Yifu, Colin Wong, Diego Montes, et al. ultralytics/yolov5: V7. 0-yolov5 sota realtime instance segmentation. *Zenodo*, 2022. 3

[13] Glenn Jocher, Alex Stoken, Jirka Borovec, Ayush Chaurasia, Liu Changyu, Adam Hogan, Jan Hajek, Laurentiu Diaconu, Yonghye Kwon, Yann Defretin, et al. ultralytics/yolov5: v5. 0-yolov5-p6 1280 models, aws, supervise. ly and youtube integrations. *Zenodo*, 2021. 2

[14] Parvinder Kaur, Baljit Singh Khehra, and Er Bhupinder Singh Mavi. Data augmentation for object detection: A review. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 537–543. IEEE, 2021. 3

[15] Fahad Shahbaz Khan, Rao Muhammad Anwer, Joost Van De Weijer, Andrew D Bagdanov, Maria Vanrell, and Antonio M Lopez. Color attributes for object detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3306–3313. IEEE, 2012. 1

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *CoRR*, abs/1902.07296, 2019. 3

[18] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. Foveabox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29:7389–7398, 2020. 1

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 4

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 2

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 1

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

[25] Zili Liu, Tu Zheng, Guodong Xu, Zheng Yang, Haifeng Liu, and Deng Cai. Training-time-friendly network for real-time object detection. In *proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11685–11692, 2020. 2

[26] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 2022. 3

[27] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: towards balanced learning for object detection. *CoRR*, abs/1904.02701, 2019. 1

[28] Youwei Pang, Xiaoqi Zhao, Lihe Zhang, and Huchuan Lu. Multi-scale interactive network for salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[29] Anima Pramanik, Sankar K Pal, Jhareswar Maiti, and Pabitra Mitra. Granulated rcnn and multi-class deep sort for multi-object detection and tracking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(1):171–181, 2021. 1

[30] Joseph Redmon. Darknet: Open source neural networks in C. http://pjreddie.com/darknet/, 2013–2016. 2

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[32] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 2

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 1

[35] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 3

[36] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[37] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, and Francesc Moreno-Noguer. Fracking deep convolutional image descriptors. *CoRR*, abs/1412.6537, 2014. 1

[38] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 1, 2

[39] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 1, 3

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[41] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001. 1

[42] Kailas Vodrahalli, Ke Li, and Jitendra Malik. Are all training examples created equal? an empirical study. *arXiv preprint arXiv:1811.12569*, 2018. 5

[43] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020. 2

[44] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *proceedings of the IEEE/CVF international conference on computer vision*, pages 9197–9206, 2019. 2

[45] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14408–14419, 2023. 1, 3

[46] Mingle Xu, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, page 109347, 2023. 3

[47] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2