– Independent Work Report Spring 2022 –

Score Calibration using Common Patterns in Object Detection

Zishuo Zhang
Adviser: Prof. Olga Russakovsky

Abstract

Although deep-learning based object detectors vary greatly in terms of size, speed and architecture, few patterns are commonly observed across most modern detection models: (1) small objects are difficult to detect, due to their small size that offers limited amount of useful information, (2) tail classes perform poorly on imbalanced datasets, and (3) incorporating contextual information improves detection performance. In response to these common patterns, most prior efforts have focused on the training stage, developing novel, complex model features and architectures. However, few had focused on the post-processing stage. In this paper, we explore the feasibility of post-hoc score calibration techniques using information on the size, class frequency, and context of an object. We perform experiments with hand-engineered calibration functions, as well as functions learned from neural networks. Our proposed calibration frameworks are able to make slight improvements to the mean Average Precision (mAP) metric without modifications to the detection model.

1. Introduction

Object detectors have advanced from using elementary hand-engineered features to convolutional neural networks, feature pyramids, and recently transformers, yet despite the differences between model architectures, the behavior of detectors still follow some common patterns. For instance, models usually perform poorly on small objects and under class-imbalanced settings, and incorporating contextual information can help generate more accurate classifications. Previous efforts that makes use of these common patterns have mainly focused on amending models during the model-training phase, introducing new architectures, algorithms, and loss functions for more accurate or faster detections. Our work, on the other hand, examines the prospect of boosting model performance during or after the model inference phase through confidence score calibration.

To improve accuracy of the image classification task under class imbalanced settings, such as the LVIS detection task [8], confidence score calibration serves as an effective way to mitigate bias against tail classes by scaling up scores of tail classes and scaling down scores of head classes [2, 19]. Since classification is part of the detection task, tail classes with few training examples also perform poorly on object detection and segmentation models, and [21, 5, 26] showed that confidence score calibrations can also be applied to detection models for AP improvements of tail classes.

In addition to class-imbalanced learning, confidence score calibration exhibits the potential to correct for data imbalances in object size. In COCO [16], less than 5% of object pixels are dedicated to small objects, which are defined to be less than 32^2 pixels, presenting a significant imbalance of training information with regards to object size; in addition, smaller objects are more likely to be missed or misclassified than larger objects. Failure on small objects on the COCO dataset [16] can be partially attributed to low recall, because the confidence scores on small objects are, on average, much lower than their medium and large counterparts, thus even correct small objects are not recalled with high scores. Therefore, using object size as a cue for score calibration has the potential to improve AP by recovering many false negatives.

Confidence score calibration can also be used to incorporate contextual information in object detection. Datasets like COCO that are designed to mimic real-life object distributions usually comes with inherent contextual bias, which describes the frequent co-occurrences between certain object classes in the same scene. We can utilize this contextual bias to increase the scores of objects that are in-context, since its context serves as a cue to signal that there is a higher probability for a correct detection. For example, a detection for "microwave" is more likely to be correct if it is discovered next to an oven, because they are usually adjacent in kitchens in the real world.

Using patterns and intuitions provided by class frequencies, object sizes, and contextual bias, we perform score calibration through hand-engineered functions and neural networks. Our calibration pipelines are able to improve performance on the mAP benchmark on LVIS and COCO for a Faster-RCNN baseline.

2. Background and Related Work

2.1. Object Detection

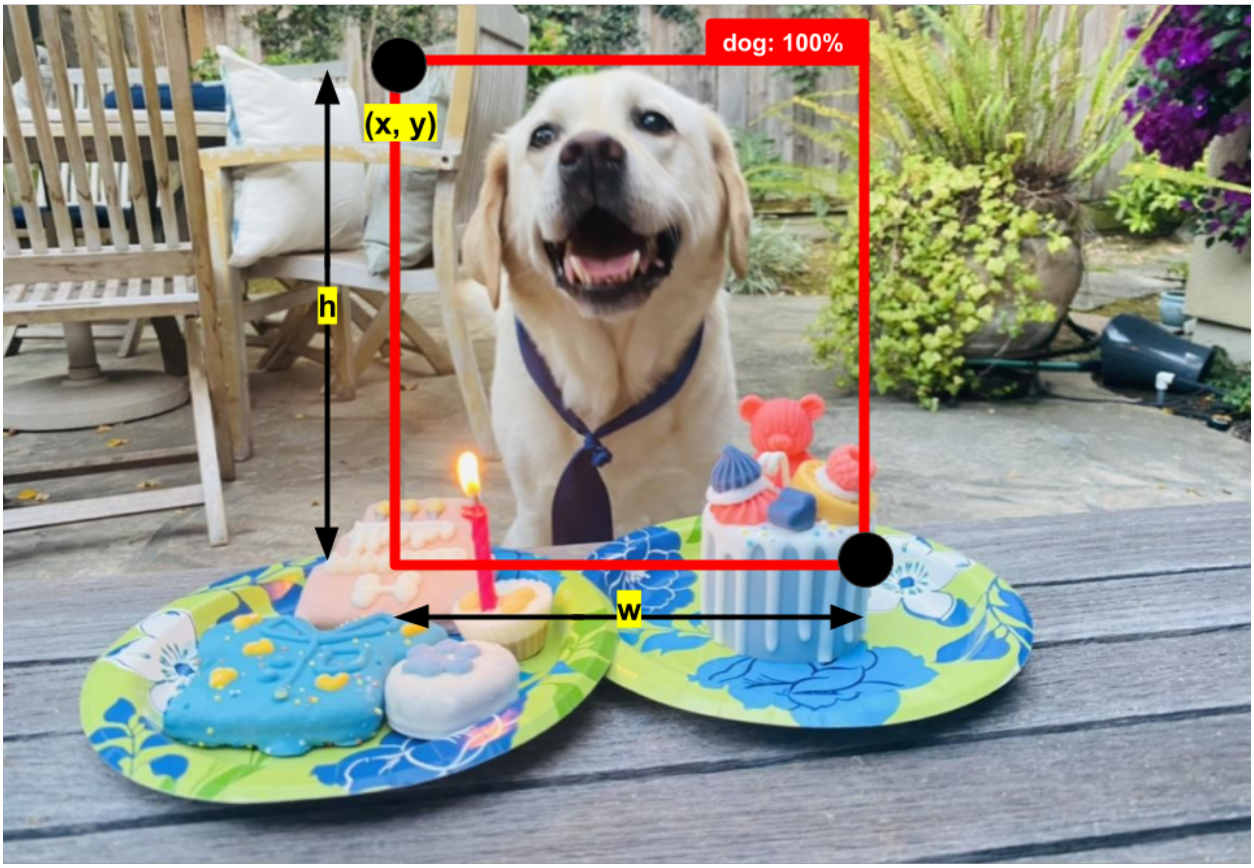

Figure 1: An example of object detection result. The detector discovered a dog with top left box coordinate (x, y), height h, width w classification of dog, and a confidence score of 100%.

The task of object detection consists of two components: localization and classification. Given an input image, an object detector should output a bounding box, a classification, and a confidence score for each object it uncovers. Formally, the output for each object can be described in the format of $(x, y, w, h, \text{class}, \text{score})$. Here, (x, y) refers to the coordinate for the top left corner of the bounding box; w, h are the width and height of the bounding box, respectively; class is the classification of the object; and score denotes the confidence score, or the probability that an object is present in the bounding box. An illustration of an example output is shown in Figure 1.

2.2. Class Imbalance in LVIS

The Large Vocabulary Instance Segmentation dataset (LVIS) is a dataset designed for long-tail object recognition, containing the same 100,700 images from the COCO dataset, but re-labels each image and constructs a set of 1,270,141 training instances over 1230 object classes (as opposed to 80 classes in COCO). In addition to tagging objects by their size (small, medium, and large), LVIS also tags them by their frequencies (rare, common, and frequent). Rare object classes appear in less than 10 images, common object classes appear in between 10 to 100 images, and frequent object classes appear in more than 100 images. Out of the 1230 classes in LVIS, 454 are rare, 416 are common, and 315 are frequent, hence the dataset is dominated by rare classes.

AP evaluation can be performed over object frequency tags. Namely, LVIS offers three additional AP statistics than COCO: AP_r, AP_c, AP_f, which refers to the AP for rare, common, and frequent classes, respectively. AP for rare and common classes are often much lower than frequent classes. A Faster-RCNN baseline achieves only 0.8% AP**r**. However, since AP is calculated by taking the average of AP for each class, rarer classes, although containing very few images, are weighted the same as frequent classes during AP calculation. Therefore, an extremely low score in rare classes can significantly influence the overall AP.

2.3. Small Object Detection in COCO

Although the problem of small object detection is often overlooked in literature, several methods that introduce more complex model architecture have shown notable improvements. DSSD [6] uses a deconvolution module to upsample and aggregate feature maps generated by SSD [18] to obtain higher-resolution features. Feature Pyramid Networks [14, 17, 24] builds paths between different levels of features, allowing fusion of low-level and high-level features for context aggregation. [13, 28] applies attention mechanisms to draw more focus on small objects during detection. [12] uses Generative Adversarial Network to increase the resolution of low-level feature maps.

Figure 2: (wine glass, person) is one of the top 20 contextual bias pairs from [23]. The three pictures above from COCO all have a person present when there is a wineglass in the image.

2.4. Contextual Bias in COCO

Contextual bias can be observed in many visual datasets, including COCO, due to frequent co-occurences between certain object classes. For example, the object category "skis" are usually identified in an image where the category "person" is also present. [23] developed a metric to quantitatively measure contextual bias in the COCO-stuff dataset [3], capturing the top 20 pairs of object classes (b, c) exhibiting the most contextual bias, where b denotes a class that is most biased by c. [10] replicated [23] and found slightly different bias class-pairs.

2.5. Confidence Score Calibration

Although the field of post-hoc calibration in the context of object detection had often been overlooked, many efforts were able to achieve notable gains in AP through calibrating confidence scores.

[1] decays the detection scores of all neighbors of a detection as a continuous function of their overlap with the higher scored bounding box, instead of eliminating all neighbor boxes with low scores. [29] proposes to learn a post-hoc model that estimates the performance of the original, already-trained model during inference. [9] proposes to learn a neural network that performs NMS, with no access to image features and operating solely on detection outputs (box coordinates and a confidence scores). [21] propses a calibration framework that scales the exponentiated logits of each class from a detection's logit vector by that class's frequency in the LVIS dataset. In my independent work last semester, "Taming Small Objects in Detection," I improved the AP of an Efficientdet [24] baseline by using scores of detection results from random crops of an image to adjust the scores of detections generated by the original image using a modification to the NMS stage of post-processing.

2.6. Different Goals for Score Calibration

The methods presented in the previous section all had the goal of improving the AP of object detectors. However, since AP calculation only factors in the ranking of confidence scores of across detections instead of the actual quantities of the scores, calibrated scores, or even the original scores may not be meaningful from a probablistic viewpoint. Therefore, sometimes confidence score calibration is performed with the goal of reflecting the true probability of classification instead of improving performance on various benchmarks. [11] observes that scores of boxes that are closer to the edges of images are often too low and provides a histogram binning method to recalibrate the scores. [27] provides a systematic study on confidence scores of multiple state-of-the-art object detectors and proposes a method to extrapolate a score threshold that separates true positives from false positives.

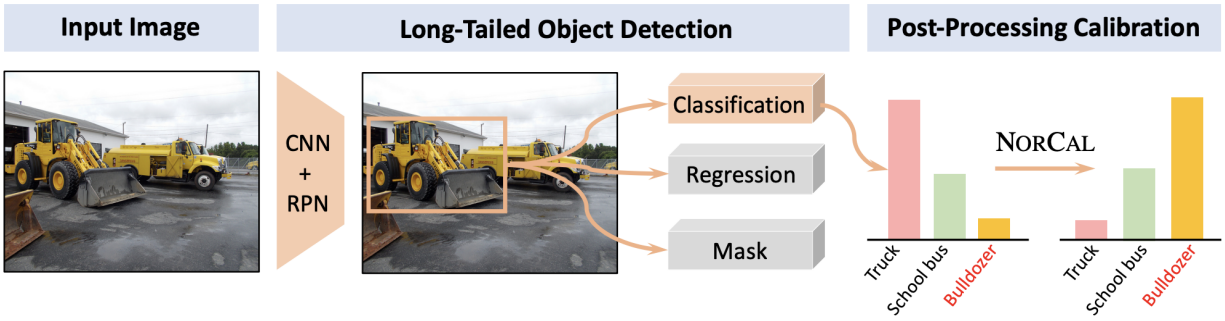

Figure 3: An illustration of the NorCal pipeline in an object detector, diagram courtesy of [21]. NorCal calibrates scores *across* classes instead of *within* classes, as shown by the rightmost diagram.

3. Methods

3.1. NorCal with Different Signals on LVIS

In classification, the softmax function is often employed to transform logits to probabilities. Suppose there are C foreground object classes numbered $\{1,2,3,...,C\}$ in a dataset, and a background class numbered 0. For a given detection and a vector $x \in \mathbb{R}^{C+1}$ containing all class logits of that detection, the confidence score $p(x_j)$ of each class $j = 0,1,...,C$ can be given as:

$$p(x_j) = \frac{exp(x_j)}{\sum_{j=0}^{C} exp(x_j)} \tag{1}$$

NorCal [21] proposed a simple calibration framework that significantly boosted the AP of a Faster R-CNN baseline trained on the LVIS0.5 dataset. Specifically, it provided an gain of about +8% AP on rare objects. NorCal calibrates the scores of foreground classes by scaling the $exp(x_j)$ term for each class j using an adjustment factor a_j, which is the number of images in LVIS that class j can be found in, also known as the image count:

$$p(x_j) = \frac{exp(x_j)/a_j^{\gamma}}{exp(x_0) + \sum_{j=1}^{C} exp(x_j)/a_j^{\gamma}} \tag{2}$$

gamma is a hyperparameter that tunes the strength of calibration. This calibration is performed before the Nonmaximum Suppression (NMS) procedure that greedily removes overlapping bounding boxes.

For detectors that use multiple binary sigmoid classifiers, such as RetinaNet [15] and EfficientDet [24], there is no background class, and the probabilities of the foreground classes need not to sum to 1. Therefore, NorCal proposes a different calibration function for these detectors:

$$p(x_j) = \frac{exp(x_j)/a_j^{\gamma}}{1 + exp(x_j)/a_j^{\gamma}} \tag{3}$$

The goal of score calibration in LVIS is to account for class imbalance, and NorCal accomplishes this by scaling up scores of tail classes that appear with less frequency. While setting a_j to image count yielded promising results, image count may not be the most optimal signal to use as a_j for score calibration; several more factors may be able to better account for class imbalance, and we propose to improve the performance of NorCal by replacing a_j with other intuitive measures of class frequency on LVIS. More specifically, classes are not solely imbalanced on image counts; they are also imbalanced on instance counts, or the number of instances of a certain class in the dataset. On a lower level, since each pixel in an object's bounding box provides some information to that object. To reflect on the imbalance of training information across classes, we can also calibrate scores of each class by the area count, which denotes the number of pixels taken up by bounding boxes containing objects of that class.

3.2. Direct Score Scaling on COCO

The most naive approach when it comes to score calibration is to directly scale the scores with some factor that relates to some attirbute of the detection. For this study, the attribute can be the size of the object or a binary indicator for the presence of contextual bias. Mathematically, given a detection i, its score s_i, and some associated attribute ϕ_i we define the new calibrated score $f(i)$ as:

$$f(i) = s_i \cdot g(\phi_i) \tag{4}$$

where we define $g(\phi_i)$ as the adjustment factor.

However, one problem with this approach is that confidence scores should fall in a range between

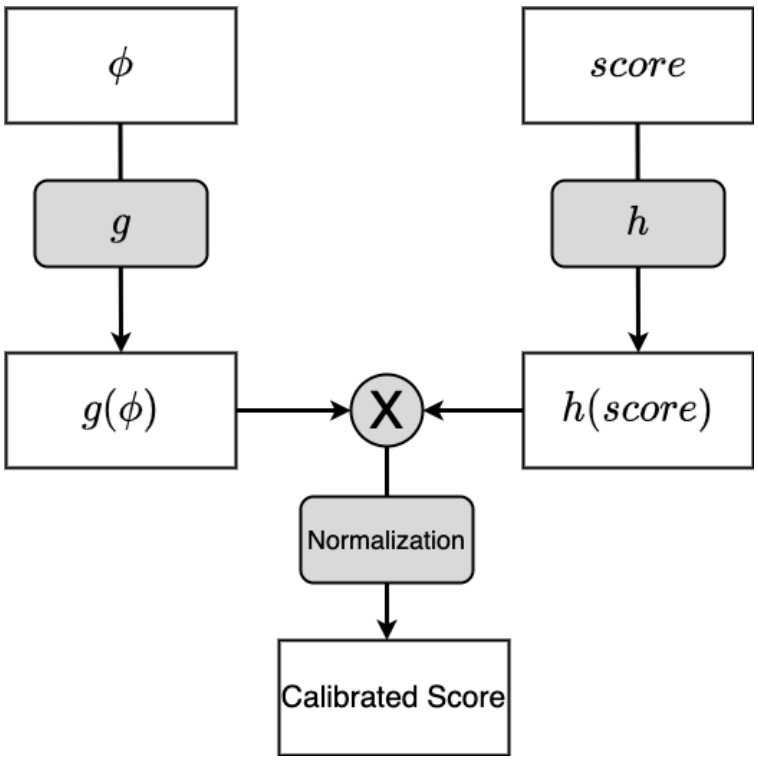

Figure 4: General calibration pipeline for direct score scaling. In our proposed calibration network, g represents the fully connected layers.

[0, 1] by definition, since they are probabilities. While it is easy to ensure nonnegative adjustment factors (e.g. taking its absolute value), there is no good way to normalize the factor itself such that $f(i) \leq 1$. We experiment with several work-arounds to normalize the range of calibrated score to [0, 1]:

1. **Max Normalization:** Divide each scores by the maximum of all detections. This is not the most natural workaround, but it ensures that the top-scoring detection receives a score of 1. Note that max normalization would not change the AP obtained from applying Equation 4, since the ordering of scores will be preserved.
2. **Partial Sigmoid Normalization:** Multiply the adjustment factor by the exponentiated logit instead of the scores themselves. Notice that since each score represents a probability, it can be written into the sigmoidal form $\frac{1}{1+e^{-x}}$, where x is the logit corresponding to the score.

Formalizing this normalization function gives us:

$$f(i) = \frac{1}{1 + \frac{1-s_i}{s_i} \cdot g(\phi_i)} \tag{5}$$

3. **NorCal Normalization:** Inspired by [21], we adopt the normalization technique for binary classifiers from NorCal for each detection i:

$$f(i) = \frac{exp(x_i) \cdot g(\phi)}{1 + exp(x_i) \cdot g(\phi)} = \frac{\frac{s_i}{1-s_i} \cdot g(\phi)}{1 + \frac{s_i}{1-s_i} \cdot g(\phi)} \tag{6}$$

In the summary, score calibration by direct score scaling can be written in the general form:

$$f(i) = N(h(s_i) \cdot g(\phi_i)) \tag{7}$$

where h, g are some transformations applied to the score s_i and its attributes ϕ_i, and N is a normalization function that ensures the value of new scores are between [0, 1]. For example, partial sigmoid normalization (Equation 5) can be rewritten with $N(x) = \frac{1}{1+x}, h(s) = \frac{1-s}{s}$. and NorCal calibration (Equation 7) can be rewritten with $N(x) = \frac{x}{1+x}$, $h(s) = \frac{s}{1-s}$. We show results of direct score scaling by object size and contextual bias under different choices of transformations h and score normalization techniques N in sections 4.3, 4.4. An illustration of the general pipeline of direct score scaling is shown in Figure 4.

Our score calibration techniques is different from previous attempts like NorCal in two aspects:

1. Our calibration is performed *post*-NMS, operating on a much limited list of detection than procedures performed *pre*-NMS. This distinction is illustrated in Figure 5
2. We calibrate scores *within* classes, instead of *across* classes. Since we are operating post-NMS, we only have access to the score of the top-scoring class instead of all classes. Thereofore, unlike NorCal, our calibration would not change the detections' final class label.

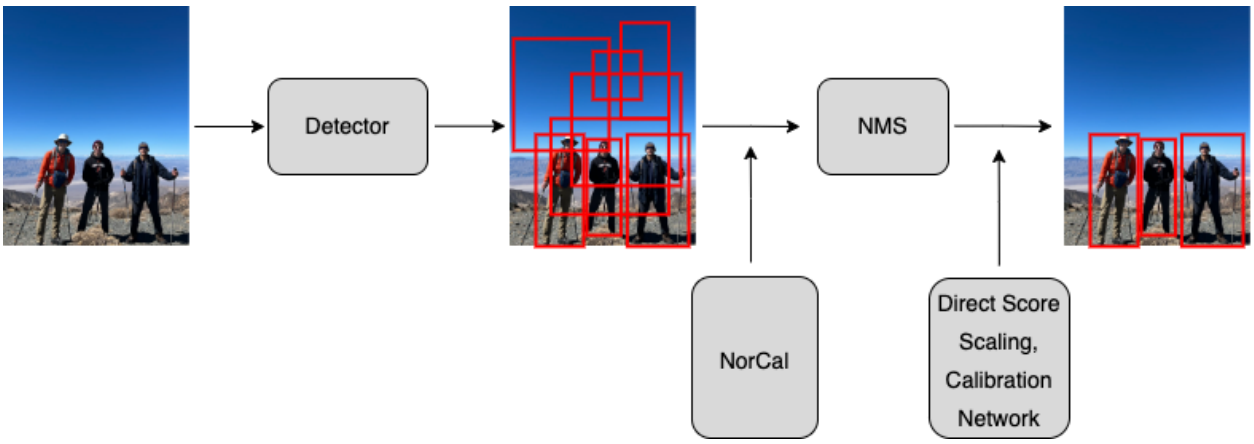

Figure 5: An illustration of the distinction between our score scaling methods and prior calibration methods like NorCal. Direct score scaling and calibration networks are both applied post-NMS, while NorCal is applied pre-NMS.

3.3. Calibration Networks on COCO

Direct score scaling and NorCal both use hand-engineered features that may require tuning of hyperparameters and exhaustive search of the transformation g to obtain the adjustment factor $g(\phi)$, which is difficult and may not yield in the most optimal results. Instead, we can train a small neural network to learn g given attributes ϕ using detection results only. In addition, learning g instead of hand-engineering it allows us to utilize information from multiple attribute at once and to capture relationships between the attributes. For example, as we find in later experiments, calibrating contextually biased detection that are large is more effective than calibrating every contextually biased detection.

The network takes as input the following four attributes from each detection:

1. **Absolute area:** the area of the detection bounding box's width multiplied by its height. This attribute is also used in the COCO metric to separate small, medium, and large objects. Small objects often have low confidence scores on COCO; calibration based on object size offers the potential to increase the recall of correct small detections. However, according to Figure 6, the majority of small objects are false positives, which is why we obtain more accurate adjustment factors using other attributes through a learnable network.
2. **Relative area:** The absolute area of a bounding box divided by the total area of the image it belongs to before the image is scaled or processed. In most object detectors, images are either

upscaled or downscaled to a fixed dimension before training. Therefore, "small" objects that also appear in small images may be upscaled to become "large" object during training.

3. **Relative Bounding Box:** [11] has shown that the location and dimension of bounding boxes negatively influence confidence scores, especially for objects closer to the edge of the image, providing room for calibration. Since training and testing images all vary in dimensions and sizes, using direct box coordinates without normalization does not provide any meaningful signal. Similar to the idea of relative area, the relative bounding box captures the position and dimension of a bounding box relative to the image it belongs to. Specifically, given the top-left corner of a box $[x, y]$, the dimension of the box $[w, h]$, and the dimension of the box's image $[W, H]$, the relative bounding box can be written as

$$[x_{rel}, y_{rel}, w_{rel}, h_{rel}] = [\frac{x}{W}, \frac{y}{H}, \frac{w}{W}, \frac{h}{H}]$$

.

4. **Contextual Bias:** A binary indicator for contextual bias. An object is contextually biased if it is a b-class in the top 20 pairs of contexutally biased classes on the COCO-stuff dataset from [10], and its corresopnding c-class is a large detection ($> 96^2$ pixels) in the same image with confidence score greater than 0.5. Refer to the definition of contextual bias in Section 4.4. We hypothesize that contextually biased objects are more likely to be correct due to the presence of a contextual cue.

Inputs are fed into a series of fully connected layers to learn an adjustment factor, which we can then use to calibrate scores following a similar approach as the direct score scaling method. In fact, the mathematical formulation of the neural network is very similar to direct score scaling, since the goal is to learn the adjustment factor $g(\phi)$, and g is simply the fully-connected layers in our network. The calibration network is trained using the binary cross-entropy loss.

Since the network we're training is a classification network, one question we need to consider is how do we obtain the ground truth labels for each detection? There are often more detections

than ground truth objects, so we need to assign each detection a ground truth label through some matching algorithm. The two algorithms commonly used for the matching detection and ground truths are described as below:

1. **Region Similarity:** This approach is usually used to match object anchors to ground truths during the training process. For our purposes, we simply match each detection with the ground truth that has the highest IoU. If the highest IoU is greater than 0.5, then we consider this pair as a match. A ground truth may be matched to multiple detections, as long as its IoU with its matches are greater than 0.5. If a detection is unmatched or matched with a ground truth from a different class, then we label the detection as a false positive; otherwise, the ground truth is labeled as a true positive.
2. **Per-class Matching:** This approach is performed during COCO's AP evaluation which takes place post-NMS. Given a class, this algorithm greedily matches all detections of that class and all ground truth with that class that meets a certain IoU trehshold (0.5 for our experiment). In this approach, there is a one-to-one relationship between detection and ground truth matches. If the detection is unmatched, then it is a false positive; if matched, then it is a true positive.

We compare performance of our network using the two matching algorithms, different architectures of the fully connected layers, and normalization functions in section 4.5.

4. Experiments

4.1. Experimental Setup

Data: For testing NorCal using different signals for long-tail distributions, we use the LVIS0.5 validation dataset, which consists of the same images as COCO val2017 but different annotations. For testing direct score scaling and calibration networks, we use detection results of 5000 images from COCO val2017. For neural network methods that requires training data, we adapt a three-fold cross validation strategy with 2:1 train:test split on the COCO val217 dataset.

Baseline: The baseline model we use is an FPN-based Faster-RCNN [22, 14] with a ResNet-50

backbone. pretrained on COCO's training2017. We obtain the pretrained model from the Detectron Model Zoo [7]. The model achieves a baseline AP of 40.2% on COCO val2017.

4.2. NorCal with Different Signals

Attribute (ϕ)	γ	**AP**	**AP$_s$**	**AP$_m$**	**AP$_l$**	**AP$_r$**	**AP$_c$**	**AP$_f$**
Baseline	Baseline	23.6	18.7	28.7	37.4	14.2	22.3	29
image count	0.7	26.2	20.5	31.5	40.1	**18.5**	**26.1**	29.3
area count	0.3	26	20.2	31.4	**41.1**	17	25.6	**30**
instance count	0.5	26.3	20.9	**31.7**	40.9	17.7	26	29.9
log(area count)	1	**26.5**	**21**	31.4	41	18.1	25.8	29.7

Table 1: Results from NorCal using different signals that accounts for class frequency in the LVIS dataset.

We report the results of NorCal using different measures of object frequency in the LVIS dataset. Concretely, we replace a_j in Equation 2 , which is the image count of an object class j in the original NorCal implementation. The candidates of other possible a_j's in our experiments includes (1) instance count and (2) area count. Since the number of pixels of certain classes is orders of magnitudes larger than the instance count or image count, we also experiment with (3) logarithm of area count. The hyperparameter γ in Equation 2 is tuned on a subset of 5000 training images through grid search as noted by the original authors of [21].

We find that setting $a_j = \log(\text{area count})$ with a $\gamma = 1$ yielded the most improvement in overall box AP (+2.9% from baseline, +0.3% from image count), followed by instance count (+2.7% from baseline, +0.2% from image count). This improvement confirms our hypothesis that the area count of the class gives a better signal for NorCal calibration, as it is a better representation of the amount of training information for a particular class.

4.3. Direct Score Scaling by Object Size

Recall equation 4; in the scenario of scaling by object size, $g(\phi) = \frac{1}{\text{absolute area}}$. As noted before, since detectors like Faster-RCNN rescales images to the same dimension before training, we also experiment direct score scaling with the relative area of a detection. Since the absolute area of

Adjustment Factor ($g(\phi)$)	Normalization	AP	AP_s	AP_m	AP_l
Baseline	Baseline	40.2%	24.2%	43.5%	52.0%
1 / Absolute Area	Divide by Max	26.0%	**27.7%**	25.2%	43.3%
1 / Absolute Area	NorCal	36.9%	25.7%	**45.4%**	**53.2%**
1 / Relative Area	Divide by Max	14.7%	11.5%	21.3%	28.9%
1 / Relative Area	NorCal	32.8%	20.0%	38.9%	47.9%
1 / log(Absolute Area)	Divide by Max	38.1%	25.0%	44.7%	52.2%
1 / log(Absolute Area)	NorCal	**40.2%**	24.5%	43.8%	52.2%
1 / sqrt(Relative Area)	Divide by Max	21.7%	16.1%	28.8%	36.5%
1 / sqrt(Relative Area)	NorCal	37.5%	22.0%	41.1%	49.9%

Table 2: Results from direct score calibration using different attributes ϕ and normalization functions as defined in Section 3.2

.

objects may be very large, which leads to unstability of the calibration, we also experiment with $g(\phi) = \frac{1}{log(\text{absolute area})}$ to weaken the calibration strength. Similarly, since the relative area of objects are small (between [0, 1], especially for small objects, we experiment with $g(\phi) = \frac{1}{\sqrt{\text{relative area}}}$.

We find that regardless of the transformations g or the normalization functions, the overall AP performance did not improve. However, in several cases, the sub-categories of AP, namely AP_s, AP_m, and AP_l, exhibited notable increase as shown by Table **??**. AP_s improved the most (+3.5%) with max normalization and $g(\phi) = \frac{1}{\text{absolute area}}$; AP_m improved the most with NorCal normalization and $g(\phi) = \frac{1}{\text{absolute area}}$ (+2.1%); AP_l also improved the most with NorCal normalization and $g(\phi) = \frac{1}{\text{absolute area}}$.

For the choice of $g(\phi)$, we find that using absolute area directly resulted in more gains AP_s, AP_m, and AP_l, but at the cost of larger overall AP loss. On the other hand, $g(\phi) = \frac{1}{log(\text{absolute area})}$ provides less improvement to AP_s, AP_m, and AP_l, but also with less impact in overall AP, since it is a weaker signal.

For the choice of the normalization function, we find NorCal normalization to be the most effective at balancing making improvements to AP_s, AP_m, and AP_l while preserving the overall AP. However, max normalization was able to make larger improvements on AP_s. Also we find that there is no distinction between the results of NorCal normalization and partial sigmoid normalization, because the functions $m(x) = \frac{1}{1+(x \cdot a)}$ and $n(x) = \frac{x \cdot a}{1+x \cdot a}$ share the same monotonacity, i.e. $m(x) < m(y)$ if

and only if $n(x) < n(y)$, $\forall x, y \in \mathbb{R}$. However, the two normalization functions do influence the behavior of the calibration network, which will be discussed in Section 4.5.

4.4. Direct Score Scaling by Contextual Bias

b-class	c-class	AP Gain	AP_s Gain	AP_m Gain	AP_l Gain
skis	person	**+0.21%**	0.00%	+1.80%	+10.72%
keyboard	mouse	**+0.17%**	0.00%	0.00%	+0.40%
skateboard	person	**+0.16%**	0.00%	+0.38%	-1.24%
snowboard	person	**+0.13%**	0.00%	+0.96%	-0.07%
mouse	keyboard	**+0.11%**	0.00%	0.00%	+0.06%
tennis racket	person	**+0.10%**	+0.04%	+0.40%	+1.35%
baseball glove	person	**+0.09%**	0.00%	+0.06%	0.00%
remote	person	**+0.07%**	0.00%	+0.09%	+0.60%
potted plant	vase	**+0.04%**	0.00%	0.00%	+0.34%
fork	dining table	**+0.04%**	0.00%	+0.25%	+0.05%
sports ball	person	**+0.02%**	0.00%	0.00%	-0.05%
handbag	person	**+0.01%**	0.00%	+0.13%	-0.44%
wine glass	person	0.00%	0.00%	0.00%	0.00%
microwave	oven	-0.00%	0.00%	0.00%	0.00%
cup	dining table	-0.02%	0.00%	+0.04%	-0.33%
spoon	bowl	-0.09%	0.00%	0.00%	-0.23%

Table 3: AP gains from direct score scaling using contextual bias, with $g(\phi) = 2$. b-classes are contextually biased classes; c-classes are the associated bias cues; and AP-Gains refers to AP gains in b-classes.

We display the results of the per-class AP performance on the most contextually biased *b*-classes from [10]. In the scenario of scaling by contextual bias, ϕ is a binary indicator for contextual bias. When contextual bias is present on an object, we scale its score up by a factor $g(\phi)$, which in this case is an arbitrary constant. We define contextually biased objects to satisfy the following condition:

1. Present as a *b*-class in the top 20 pairs found in [10]. [23, 10] performed experiments on the COCO-stuff dataset, but some classes from COCO-stuff are not present in the COCO train/val2017 dataset, thus we scale scores and report results on the 16 class pairs from [10] that exist in the COCO train/val2017 dataset.

2. The bounding box for the object is larger than 96^2 pixels. In other words, we calibrate only large objects. On the Faster R-CNN baseline, small and medium detections, which are mostly low-scoring, consist of a much larger number of false positives than large detections. On the other hand, the classification accuracy of large objects consists of mostly true positives, many of which have high scores, as shown by Figure 6. Therefore, re-ranking the scores from a pool of mostly incorrect detections (low-scoring small objects) higher than scores from a pool of mostly correct detections (high-scoring large objects) may harm the performance of calibration.
3. Its associated c-class is also larger than 96^2 pixels and has a score greater than 0.5. We introduce this criteria to maximize the probability that the detection for the c-class is a true positive; contextual bias cue can only be useful when the cue is actually present in the image.

b-class	**c-class**	**AP Gain,** $g(\phi) = 2$	**AP Gain,** $g(\phi) = 2.5$	**AP Gain,** $g(\phi) = 3$
skis	person	+0.21%	+0.21%	+0.28%
keyboard	mouse	+0.17%	+0.22%	+0.25%
skateboard	person	+0.16%	+0.16%	+0.11%
snowboard	person	+0.13%	+0.07%	-0.04%
mouse	keyboard	+0.11%	+0.13%	+0.09%
tennis racket	person	+0.10%	+0.10%	+0.12%
baseball glove	person	+0.09%	+0.11%	+0.12%
remote	person	+0.07%	+0.08%	+0.10%
potted plant	vase	+0.04%	+0.10%	+0.07%
fork	dining table	+0.04%	+0.04%	+0.01%
sports ball	person	+0.02%	+0.03%	+0.04%
handbag	person	+0.01%	0.00%	-0.02%
wine glass	person	0.00%	0.00%	0.00%
microwave	oven	0.00%	-0.01%	-0.02%
cup	dining table	-0.02%	-0.04%	-0.06%
spoon	bowl	-0.09%	-0.11%	-0.12%

Table 4: AP gains of direct score scaling by contextual bias using $g(\phi) = 2, 2.5, 3$. As shown, $g(\phi) = 3$ gave the strongest improvements to skis and hurt the AP of spoon the most as well.

For this calibration technique, max normalization did not yield improvements on AP, AP_s, AP_m, or AP_l. All major improvements come from NorCal Normalization and Partial Sigmoid Normalization,

both of which result in the same ordering of the scores, thus yielding the same AP. We report the performance of calibration using NorCal normalization in Table 3.

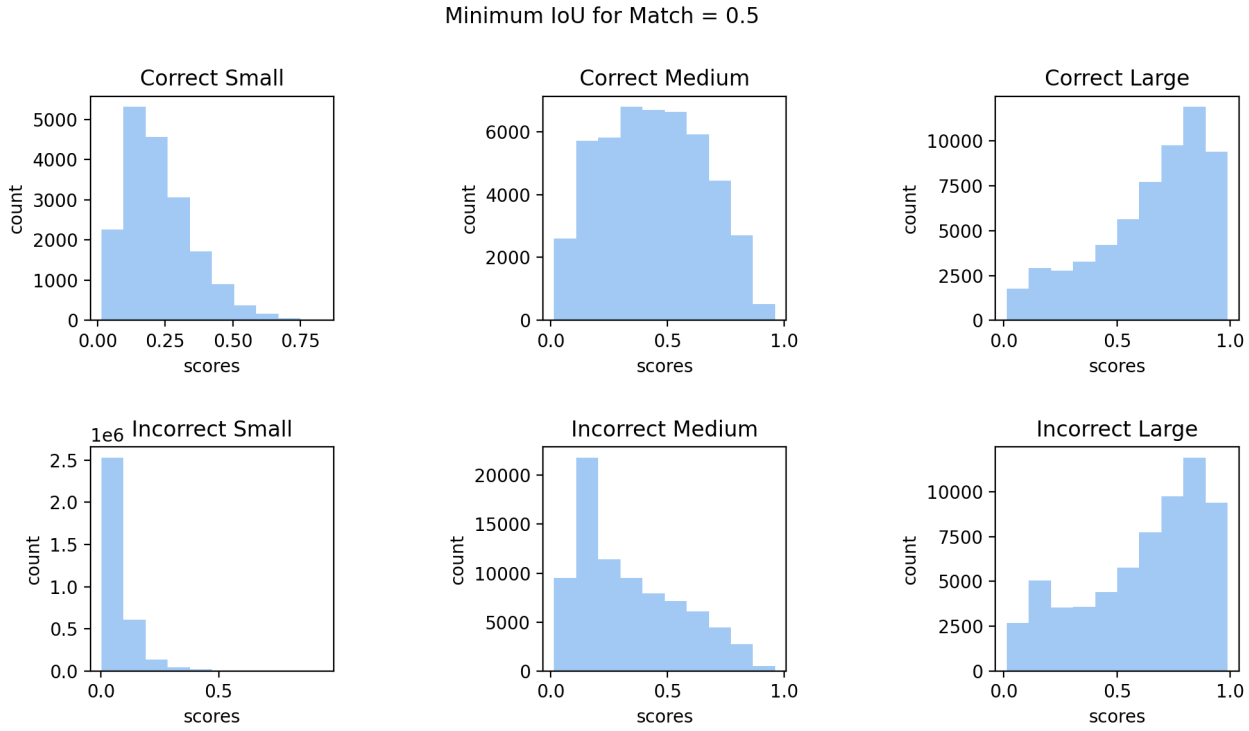

Figure 6: Distribution of scores for correct and incorrect objects, categorized by small (<32^2 pixels), medium ($> 32^2, < 96^2$ pixels), and large objects (>96^2 pixels). An object is considered "correct" if it is matched according to Per-class Matching algorithm mentioned in Section 3.3. Observe that small and medium detections consist of much more false positives than true positives, while the majority of the scores of these objects tend to be low. On the other hand, about 50 percent of large detections are true positives, and most true positives have very high scores.

Note that the choice of function $g(\phi)$, which is an arbitrary constant, also affects the performance calibration. We note that smaller $g(\phi)$ tend to bring the least amount of improvements in successful cases while costing low loss of AP in failure classes. On the other hand, larger $g(\phi)$ brings more improvements in successful classes, but with more loss of AP in failure class. Table 4 illustrates that $g(\phi) = 2$ provided the least amount of AP improvements to skis (+0.21%), but also had the least loss in AP in classes like spoon (-0.09%). On the other hand, $g(\phi) = 3$ offers +0.28% gain in skis, but also the most loss in spoons (-0.12%).

4.5. Calibration Networks

class	AP Gain	AP_s Gain	AP_m Gain	AP_l Gain
bed	+1.03%	0.00%	+2.27%	+0.98%
hair drier	+0.31%	0.00%	0.00%	+4.44%
cat	+0.21%	+1.76%	+2.53%	+2.30%
keyboard	-0.26%	+0.88%	+2.08%	+1.34%
hot dog	-0.37%	+1.80%	+1.33%	+0.36%
skateboard	-0.44%	+0.63%	+2.10%	+3.95%
dining table	-0.99%	+2.72%	+4.11%	+0.82%
pizza	-1.26%	+3.16%	+1.33%	+0.24%
dog	-1.58%	+3.81%	+1.91%	+2.07%
tennis racket	-1.67%	+2.02%	+0.93%	+2.80%
oven	-1.93%	+2.54%	+3.88%	+0.54%
surfboard	-1.99%	+0.66%	+1.14%	+2.65%
zebra	-2.08%	+0.89%	+0.68%	+0.04%
horse	-2.18%	+3.27%	+1.65%	+0.38%
carrot	-2.25%	+2.57%	+4.65%	+2.54%
elephant	-2.38%	+6.06%	+0.77%	+0.52%
snowboard	-2.42%	+1.03%	+3.69%	+1.22%
wine glass	-2.43%	+1.31%	+0.91%	+1.32%
broccoli	-2.44%	+6.57%	+2.47%	+1.49%
teddy bear	-2.48%	+2.42%	+1.81%	+0.43%
skis	-2.77%	+1.31%	+8.82%	+1.82%
sheep	-2.78%	+3.41%	+1.55%	+1.68%
airplane	-2.80%	+0.28%	+1.55%	+0.51%
spoon	-2.93%	+3.43%	+6.57%	+3.45%
fork	-2.96%	+2.95%	+3.54%	+5.39%
handbag	-3.20%	+2.02%	+4.38%	+0.76%
motorcycle	-3.23%	+1.62%	+1.55%	+1.23%
knife	-3.30%	+1.93%	+6.23%	+4.95%
bench	-3.37%	+1.10%	+1.83%	+0.73%
cup	-3.43%	+0.70%	+1.31%	+0.14%
person	-3.62%	+1.02%	+1.10%	+0.75%
chair	-3.65%	+2.84%	+2.55%	+1.22%
baseball bat	-3.66%	+3.34%	+4.68%	0.00%
remote	-3.78%	+1.34%	+3.33%	+4.13%
backpack	-3.95%	+1.43%	+3.56%	+0.55%
book	-4.02%	+1.59%	+1.63%	+1.44%
tie	-4.31%	+1.07%	+3.62%	+0.75%
sink	-4.73%	+3.14%	+1.32%	+0.62%
cake	-6.23%	+3.94%	+0.78%	+0.74%
toothbrush	-6.62%	+5.61%	+0.55%	+6.08%

Table 5: AP gains from 40 classes that showed improvements in all of AP_s, AP_m, and AP_l, after using calibration network with NorCal normalization.

We perform various experiments under different settings and configurations of the network, and examine the effects on per-category AP of changing these settings. The calibration network overall

was not able to improve AP, AP_s, AP_m, or AP_l when calculation is performed over all classes, however it was able to improve all of AP_s, AP_m, or AP_l for 40 of the 80 classes in COCO with our Faster R-CNN baseline. Detailed results of the 40 classes are shown in Table 5.

4.5.1. Training Settings We train one calibration network per class of objects. In the case of a one fully-connected layer (8 parameters), we train a total $8 \times 80 = 640$ parameters since COCO has 80 classes. The neural networks are all trained on Google Cloud TPUs with a batch size of 128, learining rate of 0.01 on the Adam Optimizer, over 5 epochs. Our Faster-RCNN baseline has 159200 detections, thus we have 159200 training samples.

4.5.2. Effect of Ground Truth Matching Algorithms We find that Region Similarity Matching, under any settings, yielded very poor results, and the AP, AP_s, AP_m, and AP_l often dropped by 10-20%. Since we are performing calibration post-NMS, it would not make sense to use a matching algorithm like Region Similarity, which is normally performed pre-NMS. In the following experiments, we assume that ground truth labels of training examples are generated via Per-class Matching.

4.5.3. Effect of Network Architecture We find through experiments that simpler architectures, especially networks with just one fully-connected layer like the one in Figure 3, yields better AP improvement. Besides AP gain, larger networks also did not offer any advantages in terms of the number of classes it was able improve. Using two fully-connected layers with the absolute value function as activation in between layers, we were only able to improve the AP_s, AP_m, and AP_l of 33 classes as opposed to 40 from the one-layer neural network setting. We chose to use the absolute value function as activation because common linear activation units like ReLU and ELU zeros out the adjustment factors when we tried them in our experiments.

4.5.4. Effect of Normalization Functions Using a simple network with one fully connected layer, We experiment with the three normalization techniques proposed for direct score scaling, and NorCal normalization was the only normalization function that was able to train data successfully without significant AP losses.

We cannot use max normalization because there is no easy way to provide the network signal on

the maximum adjusted score for a particular batch during training, meaning that scores may exceed 1 during training, which breaks the binary cross-entropy loss function used in our neural network.

While NorCal normalization and partial sigmoid normalization resulted in the same AP improvements in the case with direct score scaling, partial sigmoid normalization in the calibration network did not improve AP, AP_s, AP_m, or AP_l of any classes. We suspect that this failure comes from the fact that e^{-x} is a much weaker signal than e^{x}. Since e^{-x} can often be a very small number, it requires finer optimization of the adjustment factor $g(\phi)$ during training.

Finally, using the NorCal normalization function brings AP improvement on all of AP_s, AP_m, AP_l for 40 of the 80 classes in COCO, despite the fact that only three classes exhibited improvement on overall AP. Results from NorCal normalization on a calibration network with one fully-connected layer are shown in Table 5.

5. Conclusion

We propose three confidence score calibration methods that make use of common patterns in object detection. To resolve the issue of class imbalance, we adopt the approach taken by [21] and improve its performance by making slight adjustments to its measure of class frequency on the LVIS dataset. To mitigate failure cases on small objects and to make best use of contextual cues among detections, we propose a direct, hand-engineered score scaling function and normalization pipeline, as well as a calibration network that learns the scaling function. Through extensive experiments, we have found that our proposed approaches on confidence score calibration using object size, contextual bias, and box coordinates shows promising signals of improvement, and is especially useful in providing performance gains on per-class AP's and AP's calculated separately for small, medium, and large objects.

Going forward with the project, we can perform the following two experiments to look for more improvements:

1. Replacing the cross-entropy objective function in the calibration network with a loss function that better mimics the AP metric. Although the cross-entropy loss is a natural loss function for

the classification task, the actual quantities of the scores do not affect AP; instead, the ranking of scores between detections is the important factor that influences the shape of precision-recall curves which affects the calculation of AP. While AP is a non-differentiable metric, several works [4, 25, 20] have already proposed novel loss functions that serve as surrogates for the optimization of AP, and running experiments using these novel loss functions may yield better performance gains.

2. Currently, we are only utilizing the one confidence score directly from the detection outputs for all of our approaches. However, detectors generate a score for all classes, and the score shown in the final detection output is only the maximum of all classes. Therefore, we have un-utilized information that may potentially be helpful, especially as inputs for the calibration network.

6. Acknowledgements

I would like to thank Professor Olga Russkovsky and Sunnie Kim for their guidance on this project, and the Princeton Visual AI Lab for providing computing resources.

References

[1] N. Bodla *et al.*, "Soft-nms – improving object detection with one line of code," 2017. Available: https://arxiv.org/abs/1704.04503

[2] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, oct 2018. Available: https://doi.org/10.1016%2Fj.neunet.2018.07.011

[3] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," 2016. Available: https://arxiv.org/abs/1612.03716

[4] K. Chen *et al.*, "AP-loss for accurate one-stage object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3782–3798, nov 2021. Available: https://doi.org/10.1109%2Ftpami.2020.2991457

[5] A. Dave *et al.*, "Evaluating large-vocabulary object detectors: The devil is in the details," 2021. Available: https://arxiv.org/abs/2102.01066

[6] C.-Y. Fu *et al.*, "Dssd : Deconvolutional single shot detector," 2017.

[7] R. Girshick *et al.*, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[8] A. Gupta, P. Dollar, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," 2017. Available: https://arxiv.org/abs/1705.02950

[10] S. S. Y. Kim *et al.*, "[re] don't judge an object by its context: Learning to overcome contextual bias," 2021. Available: https://zenodo.org/record/4834352

[11] F. Kuppers *et al.*, "Multivariate confidence calibration for object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, jun 2020. Available: https://doi.org/10.1109%2Fcvprw50498.2020.00171

[12] J. Li *et al.*, "Perceptual generative adversarial networks for small object detection," 2017.
[13] J.-S. Lim *et al.*, "Small object detection using context and attention," 2019.
[14] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," 2017.
[15] T.-Y. Lin *et al.*, "Focal loss for dense object detection," 2017. Available: https://arxiv.org/abs/1708.02002
[16] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," 2015.
[17] S. Liu *et al.*, "Path aggregation network for instance segmentation," 2018.
[18] W. Liu *et al.*, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2
[19] A. K. Menon *et al.*, "Long-tail learning via logit adjustment," 2020. Available: https://arxiv.org/abs/2007.07314
[20] K. Oksuz *et al.*, "Rank sort loss for object detection and instance segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2021, pp. 2989–2998. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00300
[21] T.-Y. Pan *et al.*, "On model calibration for long-tailed object detection and instance segmentation," in *Advances in Neural Information Processing Systems*, M. Ranzato *et al.*, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 2529–2542. Available: https://proceedings.neurips.cc/paper/2021/file/14ad095ecc1c3e1b87f3c522836e9158-Paper.pdf
[22] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015. Available: https://arxiv.org/abs/1506.01497
[23] K. K. Singh *et al.*, "Don't judge an object by its context: Learning to overcome contextual bias," 2020. Available: https://arxiv.org/abs/2001.03152
[24] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.
[25] C. Tao *et al.*, "Searching parameterized AP loss for object detection," in *Advances in Neural Information Processing Systems*, A. Beygelzimer *et al.*, Eds., 2021. Available: https://openreview.net/forum?id=hLTZCN7f3M-
[26] T. Wang *et al.*, "The devil is in classification: A simple framework for long-tail object detection and instance segmentation," 2020. Available: https://arxiv.org/abs/2007.11978
[27] S. Wenkel *et al.*, "Confidence score: The forgotten dimension of object detection performance evaluation," *Sensors*, vol. 21, no. 13, 2021. Available: https://www.mdpi.com/1424-8220/21/13/4350
[28] F. Zhang *et al.*, "Multiresolution attention extractor for small object detection," 2020.
[29] X. Zhang, S. Oymak, and J. Chen, "Post-hoc models for performance estimation of machine learning inference," 2021. Available: https://arxiv.org/abs/2110.02459