– Independent Work Report Fall 2021 –

Taming Small Objects in Detection

Zishuo Zhang
Adviser: Prof. Olga Russakovsky

Abstract

For object detection on the COCO dataset, small objects are the most abundant in amount, also the most difficult to detect, leaving a large room for improvement. There has not been many past work on object detection that specifically target small objects, and those that do relied on novel, more complex model architectures to improve small object performance. In this study, we focused on the preprocessing and postprocessing stages of object detectors and developed several techniques that can be applied during these stages to improve performance on small objects. Specifically, in the prepossessing stage, we studied different configurations of the copy-pasting augmentation and found that copy-pasting small objects and downsampled medium, large objects randomly onto images yielded the best results. In the postprocessing stage, we used predictions from upsampled crops of the original image to adjust the confidence scores of detected small objects. Both methods resulted in 34.4% box AP on COCO val2017, an improvement of +0.2% from our baseline model.

1. Introduction

Object detection is a classic computer vision task that is very relevant to real world applications such as remote sensing, security surveillance, and skin cancer detection detection. Given an input image, the goal is to output the location and the classification of all objects in the image. The recent advance of deep learning allowed object detection to leverage large datasets to produce state-of-the-art results. One of the most phenomenal datasets is the Common Objects in Context dataset, otherwise known as COCO [15], and it is often used in developing most modern object detectors.

COCO separates objects into three categories: small, medium, and large, and the numerical cutoffs for each category is shown in Table 1. As illustrated in Figure 1, it is often difficult for

	Min. Area	Max. Area	Object Count	Area Occupied
Small	0	32^2	42%	5%
Medium	32^2	96^2	34%	15%
Large	96^2	∞	24%	80%

Table 1: Size ranges of different small, medium, and large objects in 2017 COCO training and validation set, as well as their proportions in terms of object count and area (number of pixels) occupied.

humans to detect small objects because of their lack of details and vulnerability for confusion with background noise. Without careful inspection, the person dawning black clothing, labeled in the pink box in Figure 1, blends in perfectly with the light pole nearby.

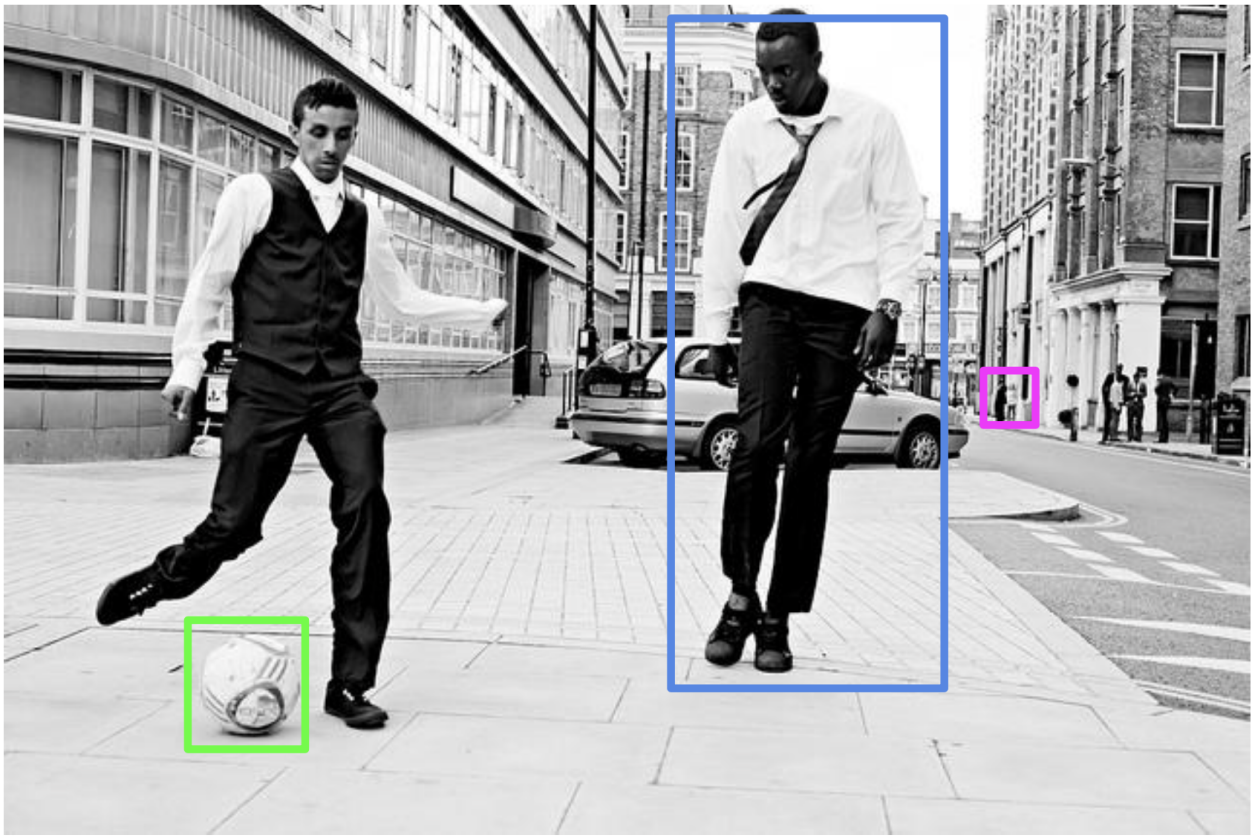


Figure 1: This is an example image from COCO. Small object is labeled in the pink bounding box, medium object labeled in the green bounding box, and large labeled in the blue bounding box.

Similarly, machines also have a challenging time detecting small objects. COCO uses a benchmark called mean Average Precision (AP) to measure object detection performance. AP can be calculated for all objects, or for small, medium, and large objects separately. For deep-learning-based object detectors trained and tested on COCO, small objects often exhibit the lowest AP. In

fact, in many state-of-the-art detectors presented in Table 2, AP for small objects is often twice or three times lower than the AP for large objects. Furthermore, we find that the COCO training and validation sets contain more small objects than medium and large objects, taking up 42% of all objects as shown in Table 1. Since AP calculation weighs objects of different sizes equally, an increase in small AP can potentially yield a noticeable increase in the overall AP.

	Small AP (AP_S)	**Medium AP (AP_M)**	**Large AP (AP_L)**
EfficientDet-D0 [19]	12.4%	39.0%	52.7%
CornerNet [11]	20.8%	44.8%	56.7%
YOLOV4 [1]	24.3%	46.1%	55.2%
SSD [17]	10.9%	31.8%	43.5%

Table 2: This table shows a comparison of small, medium, and large AP for various object detection models. All models take an input image with size 512x512.

The goal of this paper is to improve the AP for small objects in COCO. Not only will this improvement enhance an object detector's overall performance, it is also extremely relevant to areas such as autonomous vehicle perception, where small objects such as pedestrians and obstacles are especially prevalent.

One of the main challenges of detecting small objects is that COCO offers very scarce information about them. Although small objects in COCO are the most abundant in amount, they occupy the least area, taking up only 5% of all pixels that contain labeled objects, as shown in Table 1. The lack of examples makes it more challenging for models to collect information about small objects during training and test time. Prior methods that saw improvements in small object detection often focused on using more complex model architectures and modules to generate more information about small objects in the form of higher, deeper-level features.

This paper drives the attention away from detection models themselves and focus on techniques that can be applied for data preprocessing before training, and postprocessing during inference. These techniques are developed with the purpose of providing the detectors supplementary data to make up for the lack of information on small objects from the original dataset. Our contributions are outlined as follows:

1. We enhanced the copy-paste augmentation, where objects from one image are pasted onto a random location on another image before training a model, thereby giving the dataset extra examples of small objects.
2. We encourage models to include more small objects in their output by proposing a new method to adjust confidence scores of small objects using predictions from upsampled crops of the input image.

2. Background and Related Work

2.1. Object Detection

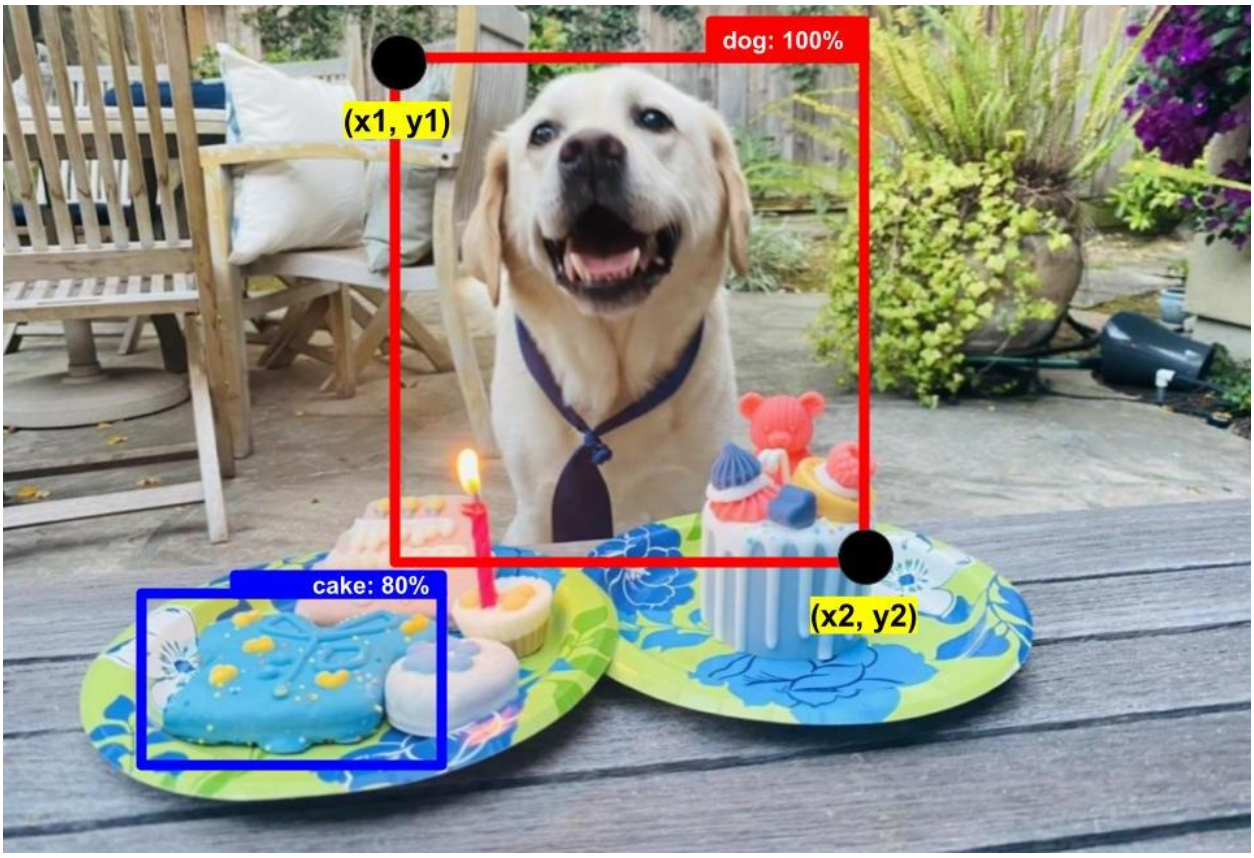


Figure 2: An example of object detection result. The detector discovered a dog with top left box coordinate (x1, y1), bottom right box coordinate (x2, y2), classification of dog, and a confidence score of 100%.

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 $(x_1, y_1, x_2, y_2, class, score)$. Here, (x_1, y_1), (x_2, y_2) are the coordinates for the top left and bottom right corner of the bounding box, respectively; *class* is the classification of the object; and *score* denotes the probability that an object is present in the bounding box. An illustration of an example output is shown in Figure 2.

2.2. Small Object Detection

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 [17] to obtain higher-resolution features. Feature Pyramid Networks [14, 16, 19] builds paths between different levels of features, allowing fusion of low-level and high-level features for context aggregation. [13, 23] 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.

Besides methods that proposed novel model architectures, several studies explored data augmentations that may aid small object detection. [10] applies techniques such as increasing image resolution, copy-pasting small objects multiple times on the same image, and oversampling small objects and yielded performance gains on all objects and small objects alone. [7] showed that randomly copy-pasting objects onto different images brings improvements to small objects as well.

2.3. Intersection over Union

An important concept in object detection is the Intersection over Union (IoU) between two bounding boxes. IoU is a ratio that is calculated by dividing the area of intersection by the area of union between two boxes. This ratio is often used to determine whether a prediction has enough overlap with the ground truth box to be considered an accurate localization of the object.

2.4. Average Precision

As introduced earlier, COCO evaluates object detection performance using a metric called mean Average Precision (AP). A higher AP denotes better performance. AP can be calculated considering all objects or for small, medium, and large objects separately. In this paper, we denote AP for small, medium, and large objects as AP_S, AP_M, and AP_L, respectively. To calculate AP, we only consider the top k highest scoring predictions, with $k = 1, 10, 100$ in most standard settings. Using these k predictions and the ground truth objects, the area under the precision-recall curve is calculated over each of the 91 object categories in COCO and over each IoU threshold ranging from 0.5 to 0.95 with a 0.05 interval. If the IoU between a ground truth box and a predicted box is below the threshold, it is considered a false positive. The average of the area under all the precision-recall curves is considered as AP in the COCO benchmark.

2.5. Measuring Errors in Object Detection

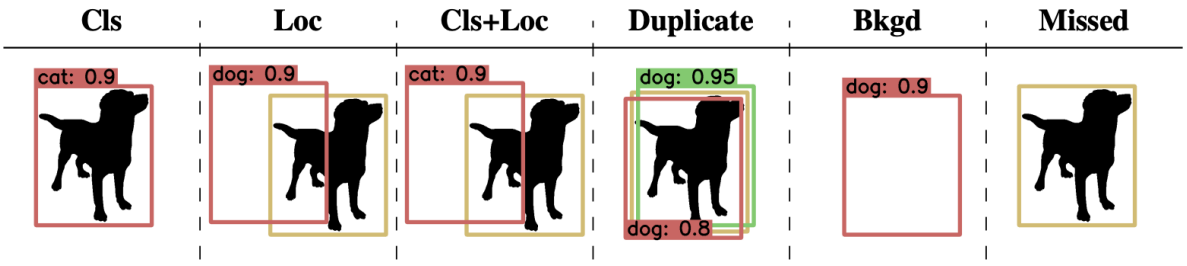


Figure 3: Visualization of different categories errors that contributes to AP loss, analyzed using the TIDE toolbox [2].

Although AP is a good way to summarize a model's performance, it does not tell us what contributes to detection errors. [9] pointed out that it is difficult to analyze the AP with respect to varying object sizes because AP is sensitive to the number of positive examples, which is different for each object size. Therefore, we cannot simply compare the AP of small and large objects directly. TIDE [2], a toolbox that identifies object detection errors, splits the loss of AP six categories, as illustrated in Figure 3. The first category is classification error (Cls), which occurs when the detector predicts the correct bounding box but the wrong class. The second category is localization

error (Loc), which occurs when the classification is correct but the location of the bounding box is considered incorrect. The third category (Cls+Loc) is when both the location of the box and the classification of the object are incorrect. The fourth category is duplicate error (Dup), when multiple boxes are overlapped with one another, but they describe the same object. The fifth category is Background error (Bkgd), when a detector places a box on the background. And finally, the sixth category is the miss error (Miss), which occurs when the detector fails to place a box around an object even though it is present in the image.

3. Methods

3.1. Inspiration

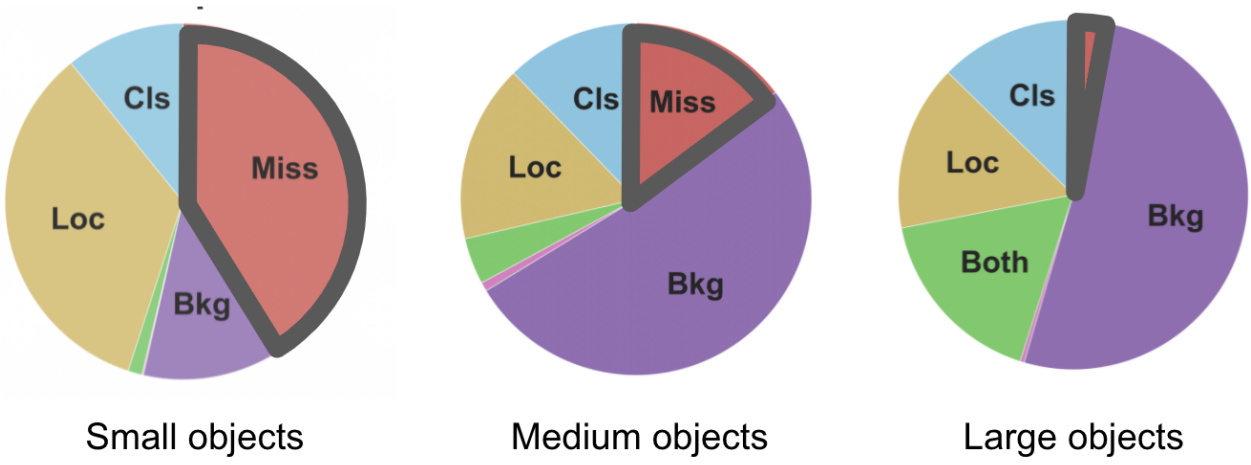


Figure 4: TIDE reveals the error composition of loss in AP_S, AP_M, AP_L. Results are obtained from running COCO val2017 on EfficientDet-D0.

To devise methods that we can use to mitigate the failure of small objects, we set up a little experiment by running our test data on several state-of-the-art models and analyzing the results using the TIDE toolbox. Shown in Figure 4 is the error decomposition of small, medium, and large objects from TIDE. The main source of error for small AP comes from having low recall, which is highlighted by the large proportion of the "missing" category from the pie chart on the left. Furthermore, we see that small objects are the most impacted by low recall, because looking at the

two charts for medium and large objects, we see that the "Miss" category contributes minimally to the overall error.

There are two possible problems that contributed to this significant loss in recall:

1. **Low confidence score:** The model was able to pick up small objects and correctly output a bounding box, but since only the top k detections with the highest confidence scores are considered in AP calculation, small objects may have been filtered out. In other word, the detector found the small objects, but with low confidence.
2. **Model incapacity:** The model was not able to pick up small objects at all. In the introduction section, we pinpointed the cause of this problem, which is simply a lack of information, meaning that models trained on the COCO dataset have not been exposed to enough examples of small objects.

Inspired by the results from the TIDE analysis, we devise three methods to address the two problems that led to low recall. To address the problem of model incapacity, we provide the model more information on small objects through copy-pasting these small objects. To address the problem of low confidence score, we propose two score adjustment methods, both using predictions from several upsampled crops of the original image.

3.2. Copy-pasting Small Objects

One of the very well-studied data augmentation method used for object detection preprocessing is the copy-paste augmentation. There're two key findings from past work. First, [10] found that copy-pasting improves the overall AP, as well as small AP, when only small objects are copied and pasted onto the same image that they belong to multiple times, as shown by the example in Figure 5.

And then, [7] has shown that copy-pasting brings effective improvement in AP, even when objects from one image is pasted onto another instead of pasting in the original image. In other words, copy-pasting is able to improve the performance of object detectors without considering the context of these objects. In addition, the performance improvement in the new study is comparable to old experiments that take context into account.

Figure 5: An example of copy-pasting from [10]. Small objects are copied and pasted on the same image multiple times. Since only small objects are pasted, the image on the bottom does not undergo any augmentation.

Figure 6: An example of copy-pasting from [7]. Objects of all sizes are copied and pasted into a new image, although they're only pasted once.

In our approach, we decide to adopt the benefits of both previous findings. We also allow copy-pasting from one image onto another, but we limit copy-pasting to only small objects to specifically target improvements of small objects. In addition, we also resize the medium and large objects to fit the criteria of small objects and paste them onto other images. In other words, we take medium and large objects and downsample them to have area less than 32x32 pixels. This allows us to gain more artificial samples and incorporate more information into our data.

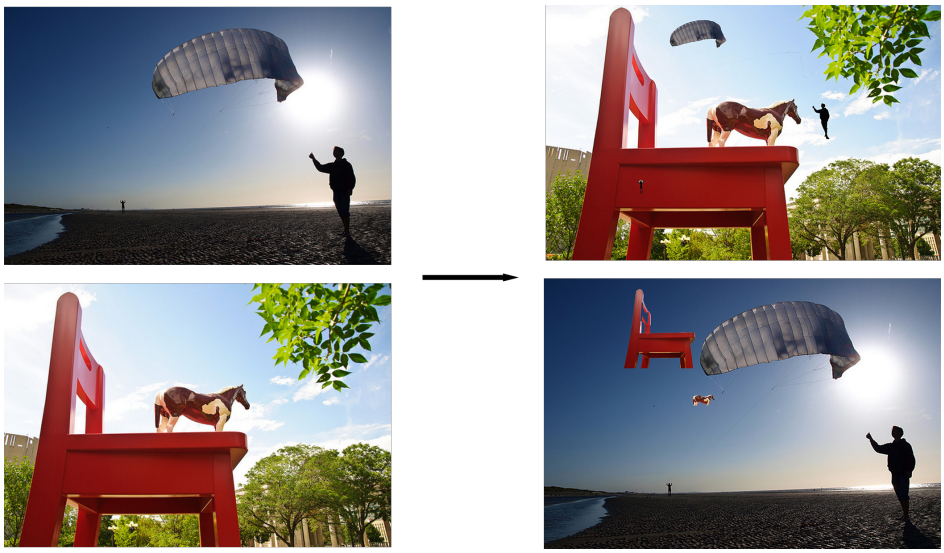


Figure 7: An example of copy-pasting from our method. Small objects are copied and pasted onto different images. Medium and large objects are downsized, then pasted as well.

Formally, given two images I_1 and I_2, where objects from I_1 are copied and pasted onto I_2, we perform copy-paste in the following manner:

1. Create three copies of I_1, $I_{1,s}, I_{1,m}, I_{1,l}$.
2. Compute three binary masks $\alpha_{1,s}, \alpha_{1,m}, \alpha_{1,l}$ using I_1, where $\alpha_{1,s}, \alpha_{1,m}, \alpha_{1,l}$ correspond to the binary mask of small, medium, and large objects in the image, respectively.
3. We downsize $I_{1,m}$ and $\alpha_{1,m}$ by a scale factor j, where j is the average area of medium objects in I_1 divided by 32^2, the cutoff area limit of small objects. This way, medium objects that are copied and pasted can be considered as small objects on average. We ignore objects downsized to less than 8x8 pixels, since they will be too difficult for machines to pick up.
4. Repeat the previous step for $I_{1,l}$ and $\alpha_{1,l}$.
5. Apply random crop and scaling to I_2.
6. Compute the final image $I_f = (1 - (\alpha_{1,s} + \alpha_{1,m} + \alpha_{1,l})I_2 + \alpha_{1,s}I_{1,s} + \alpha_{1,m}I_{1,m} + \alpha_{1,l}I_{1,l}$.

We do not apply a Gaussian filter to the masks to smooth out the edges on objects like [3], since [7] found that composing new images with and without the smoothing filter yielded similar performance.

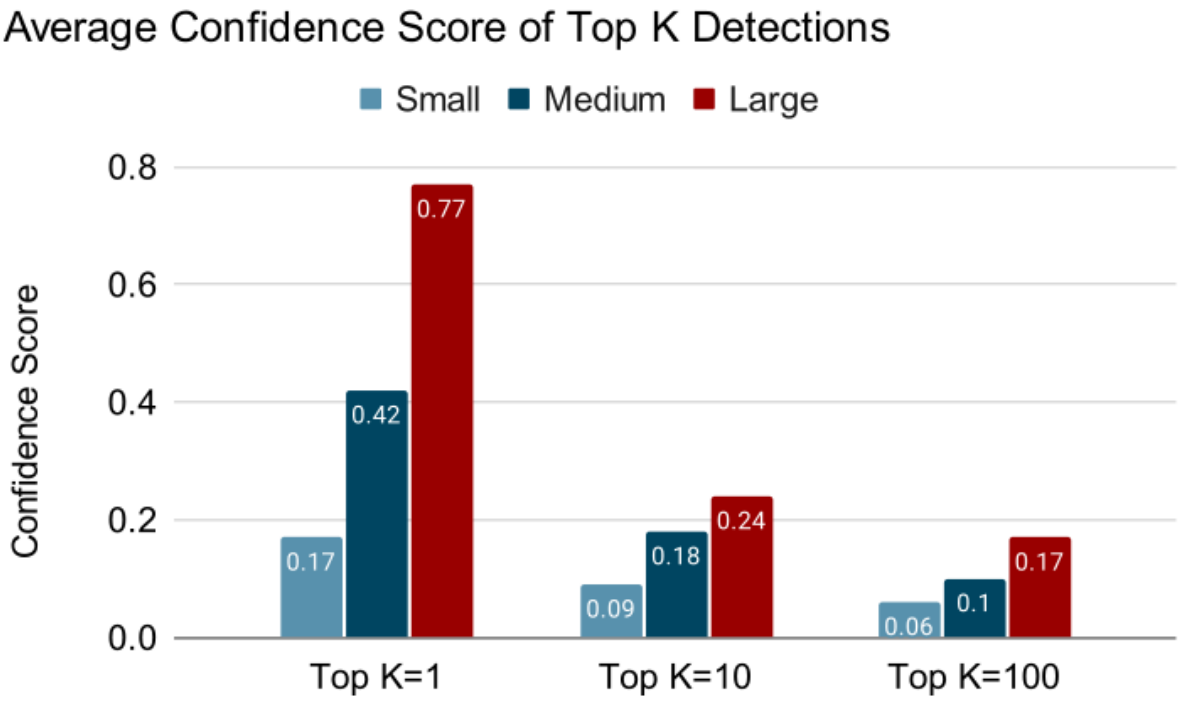


Figure 8: Confidence scores of for the top k small, medium, large prediction from running EfficientDet-D0 on COCO val2017.

3.3. Five Crop Score Adjustment

An object detector often outputs thousands of detections per image, and those with the low confidence scores are often ignored. However, COCO's AP calculation only takes into account of the top 1, top 10, and top 100 detections with the highest scores. After examining results from our baseline, EfficientDet-D0 [19], we plotted a graph showing the comparison of the average confidence scores for the top 1, 10, and 100 detections in an image. As shown in Figure 8, predictions for small objects have lower confidence scores on average than predictions for medium and large objects. This means that the final list of predictions is likely dominated by medium and large objects, whereas small objects may be filtered out, which explains why there is a gap in the proportion of "Miss" errors between small and large objects.

Given that predictions of medium and large objects often come with higher score, we can upsample different crops of the image to increase the size of small objects, such that they become effectively medium or large. Then, the models will potentially be able to produce higher scores for

these small objects.

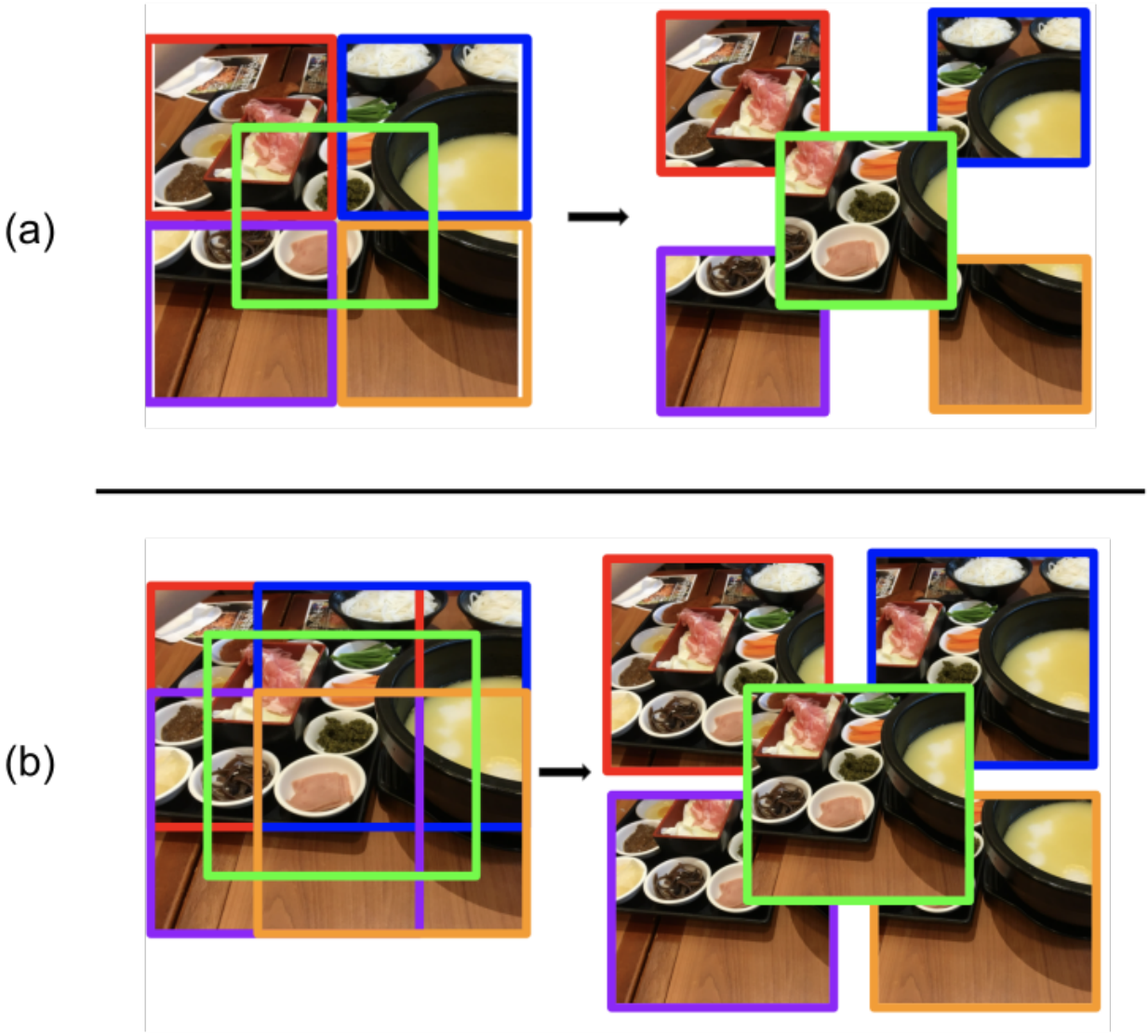


Figure 9: Two examples of how crops are formed using FiveCrop, with different scale factor m. In (a), $m = 0.5$, and in (b), $m = 0.75$.

In more detail, our method creates five crops of equal sizes for an image, one from each corner, and one in the center. The reason why we chose these five crops is that they are able to sample all parts of the images collectively, ensuring that small objects are not left out. Formally, if we adopt the standard image coordinate system with the top-left corner of the original image as $(0,0)$, and the bottom-right corner as (x,y), with x,y being the width and height of the image, respectively, then the corners of the five crops can be described as follows:

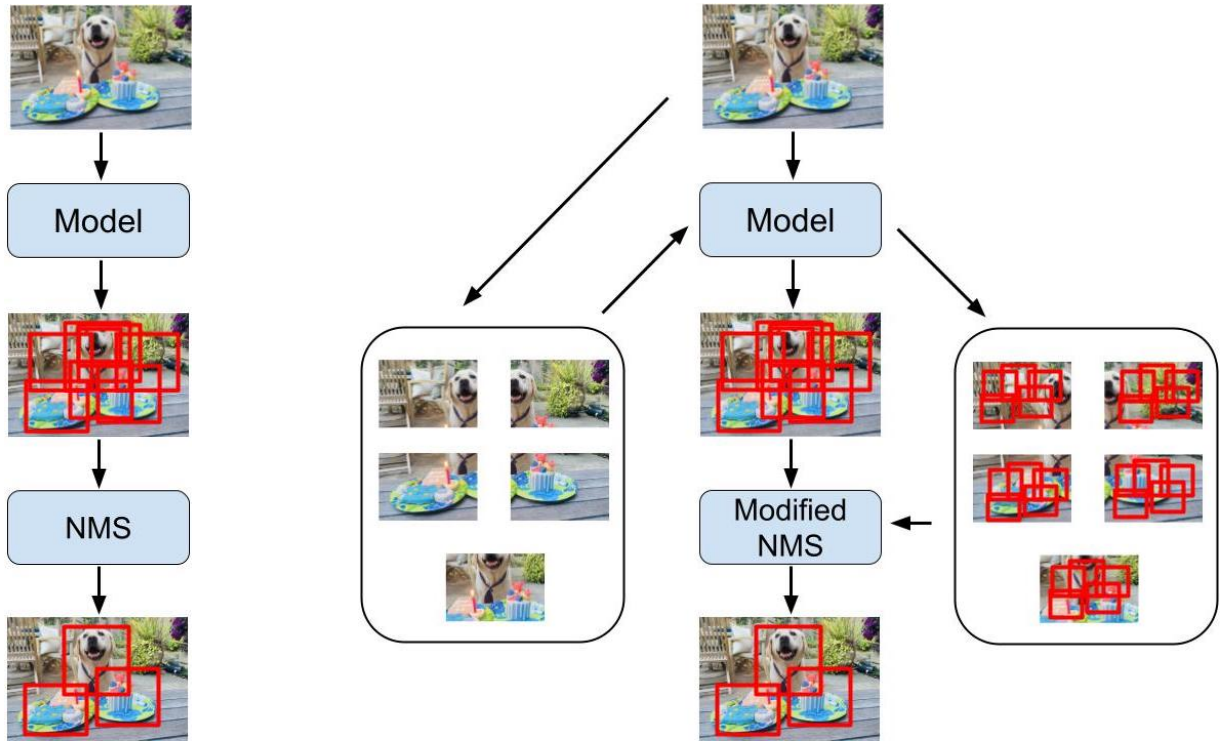
- Top-left crop: $(0,0),(mx,my)$
- Top-right crop: $((1-m)x,0),(x,my)$
- Bottom-left crop: $(0,(1-m)y,),(mx,y)$
- Bottom-right crop: $((1-m)x,(1-m)y),(x,y)$
- Middle crop: $(\frac{1-m}{2}y,\frac{1-m}{2}x),(\frac{1+m}{2}y,\frac{1+m}{2}x)$

Here, m is the scale factor that determines the size of the crop. A larger m results in larger crop size. In order for the crops to cover the entire image, $m >= 0.5$ must be satisfied. When $m = 0.5$, each crop is half the size of the original image, and when the crops are upsampled to the size of the original image, objects in the crop will be scaled twice as large as before. As m decreases, objects in the crop will become larger after upsampling. A more detailed illustration of how the crops are formed is shown in Figure 9.

Once we obtain the crops and upsample them to the size of the original image, we run model inference on the original image and each of the five crops and obtain the lists of detections. Now, since all bounding boxes in the upsampled crops are upscaled and translated, we would need to adjust the coordinates of boxes that come from crops back to what they should be in the original image.

After we have prepared and adjusted the bounding boxes for all objects, we filter out medium and large object from the crops, since we are targeting small object performance. Then, we utilize the scores from the crop predictions to adjust the scores of the predictions from the original image. To do so, we propose a method that modifies the nonmaximum suppression (NMS) algorithm in the postprocessing stage of inference. NMS is employed for pruning extra predictions from the initial list of results, since there are often many overlapping bounding boxes that describe the same object. The original NMS algorithm is as follows: Given a list of detections D, repeat until D is empty:

1. Let the detection with the highest confidence score be h
2. Add h to the final list of detections
3. Compute the IoU between h and other detections in D
4. Remove h from D



(a) NMS (b) Modified NMS

Figure 10: Comparison between model pipelines with and without using the score adjustment method. The left pipeline (a) uses normal NMS, and the right pipeline (b) uses crops and modified NMS.

5. Remove detections in D with IoU $> \gamma$ ($\gamma = 0.5$ is standard)

In our proposed modified NMS, we treat predictions from the original image and predictions from the crops separately. Let D1 be list of detections from the original image, D2 be detections from crops, the algorithm is as follows:

1. Perform original NMS on D1 and D2 separately.
2. For each h in D2:
3. (i) Compute IOU between h and all predictions in D1
   (ii) Continue if IoU $\leq \gamma$ for all predictions in D1
   (iii) Let c be prediction in D1 with the highest IoU with h
   (iv) $\text{score}_c := \max(\text{score}_c, \text{score}_h)$
4. Take D1 as the final list of predictions.

In summary, the Five Crop Score Adjustment pipeline is outlined as follows:

1. Generate five crops of equal sizes, one from each corner of the original image, and one in the center.
2. Upsample the crops to the size of the original image.
3. Run inference on the original image and each of the upsampled crops.
4. Filter out medium and large objects from the crop predictions.
5. Run modified NMS using the crop predictions and predictions from the original image.

3.4. Random Crop Score Adjustment

To give the crops more flexibility and versatility in capturing different sizes and samples of small objects, we extend the score adjustment method proposed in the last section from using five crops to using random crops as well. The idea of random cropping input images is often employed as a data augmentation before the training process in classic visual tasks such as classification and detection. Here, we select the scale factor m like in Five Crop Prediction to determine the crop size. Then, for each crop, we sample the center of the crop (p_x, p_y) from a uniform distribution. This distribution contains coordinates that lie within the rectangle defined by corners $(\frac{m}{2}x, \frac{m}{2}y), ((1-\frac{m}{2})x, (1-\frac{m}{2})y)$ to ensure that the crops do not overflow outside the image. The number of crops that we sample is a hyperparameter that should be tuned.

Just like the Five Crop method, we use prediction results from random crops in conjunction with modified NMS to adjust the scores of the detections from the original image.

4. Experiments

4.1. Experiment Setup

Data: We use 35504 images from COCO valminusminival [8] for training and 5000 images from COCO val2017 [15] for testing.

Baseline: The baseline model we use is EfficientDet-D0 [19] pretrained on COCO's training dataset shown in Figure 11. The model takes a 3-channel colored image as input, resizes it to a 512x512 image, then outputs a class and box prediction for each object it recalls. The model was

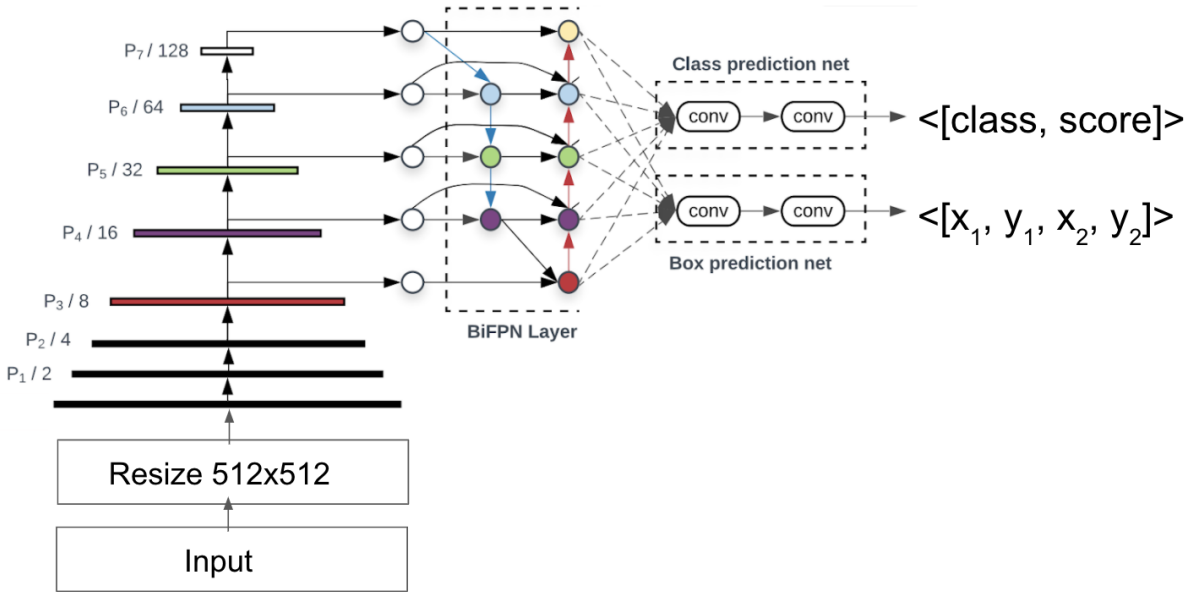


Figure 11: Basic architecture of EfficientDet [19].

trained in PyTorch by [21] and achieves a baseline AP of 34.2% and AP_S of 13.2% on COCO val2017.

4.2. Copy-pasting

In all training experiments, we use pretrained weights from EfficientDet-D0 and finetune the baseline model for 15 epochs with a batch size of 32. We compare the performance of experiments under four settings:

- Copy-pasting only small objects onto the same image they belong to, like in [10].
- Copy-pasting only small objects onto different images, like in [7], except without using medium or large objects.
- Copy-pasting all object sizes onto different images like in [7] and Figure 6.
- Copy-pasting small objects and resized medium, large objects onto different images, like in Figure 7.

The result of the experiment is shown in Table 3, where we also make comparisons to the baseline. We find that the method proposed in section 3.2, where we copy small and resized medium, large objects from one image and paste onto a different image, yielded the most increase in overall AP (+0.2%). Some other key findings include:

	AP	APS
Baseline	13.2%	34.2%
Small only, Same Image	13.5%	34.1%
Small only, Different Image	13.4%	34.1%
All Sizes, Different Image	13.4%	34.3%
Small + Resized Med/Lg, Different Image	**13.7%**	**34.4%**

Table 3: Result for copy-paste experiments.

- **Size of pasted objects does not affect AP_S.** Copy-pasting all objects results onto different images results in the same increase of AP_S as copy-pasting only small objects onto different images.
- **Overall AP decreases when only small objects are used.** This suggests that copy-pasting small objects only impedes the performance of medium and large objects.
- **Adding resized medium/large objects yields better AP_S than small objects only.** Intuitively, adding downsized medium/large objects effectively supplies the model more examples of small objects, leading to more improvements in AP_S.

4.3. Five Crop Score Adjustment

One hyperparameter for Five Crop Score Adjustment is the scale factor m that determines the size of the crops. We perform a simple grid search and found that $m = \frac{3}{4}$ worked the best for our experiments. This means that the length and width of the crops are $\frac{3}{4}$ of those from the original image.

The two most important step in the Five Crop Score Adjustment method are (1) filtering out medium and large objects from the crop predictions and (2) applying modified NMS to calibrate scores of predictions from the original image using predictions from the crops. In Table 4, we show an ablation study that compares performances of our method with and without these two important steps. The three settings of the studies is listed as follows:

1. **Concat Predictions + NMS**: Predictions from crops and the original image are concatenated then processed through regular NMS.
2. **Concat Predictions + Filter Medium/Large + NMS**: Predictions from crops and the original

image are concatenated with medium and large objects are removed from crop predictions. The concatenated list is processed through regular NMS.

3. **Concat Predictions + Filter Medium/Large + Modified NMS**: This is our proposed method. Predictions from crops and the original image are concatenated with medium and large objects are removed from crop predictions. The concatenated list is processed through modified NMS.

	AP	APS
Baseline	13.2%	34.2%
Concat Predictions + NMS	10.1%	29.3%
Concat Predictions + Filter Medium/Large + NMS	12.2%	33.7%
Separate Predictions + Filter Medium/Large + Modified NMS	**13.9%**	**34.4%**

Table 4: Results of Five Crop Score Adjustment.

As shown above, when we simply concatenate the predictions and run original NMS, we found that the performance actually dropped by a noticeable amount, and we hypothesized that there are two reasons for the failure.

The first reason is that medium and large objects are more likely to be split up due to cropping than small objects, so the boxes predicted using the crops may not be accurate coordinates that covers the entire object. However, since larger objects do tend to have higher confidence scores, the original predictions are likely filtered out because we upsampled in on the crops. The solution to this issue simple - we can filter out the boxes containing medium and large objects from the crops. Since we're targeting small objects, interference with medium and large predictions from the original image is undesired.

The second reason is that running crop prediction alone increases classification errors and falsely detects background noise, as shown by the increase in the "Cls" and "Background" category when we analyze the model with TIDE in Figure 12. Although the crops help us gain more information about the image, we also need to consider the fact that information about the context of the image might be lost when we only look at one section of an image. The loss of context may result in the detector not being able to output the most accurate class. For this reason, the five crop method

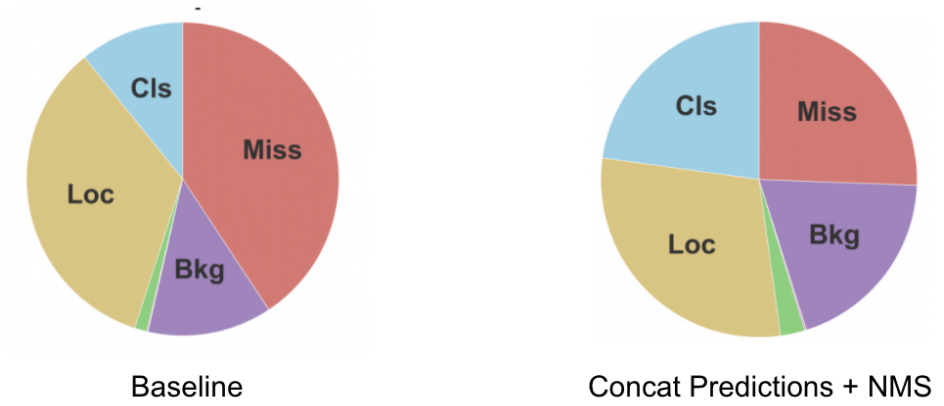


Figure 12: Visualization of errors analyzed using the TIDE toolbox.

should not be used to add new predictions to the prediction set, but to be used as a tool to adjust the scores of predictions from the original image. We perform this score adjustment by modifying the non-maximum suppression algorithm used during post-processing.

After applying all these changes, we see an improvement in both AP (+0.2%) and AP_S (+0.7%).

4.4. Random Crop Score Adjustment

For Random Crop Score Adjustment experiments, we test the effect of the number of random crops on the improvement of AP. We run our model using up to nine crops, and the crop centers are randomly sampled for every batch, with a batch size of 128. We find $m = \frac{3}{4}$ to work the best for Random Crop Score Adjustment as well. The results are shown in Figures 13, 14.

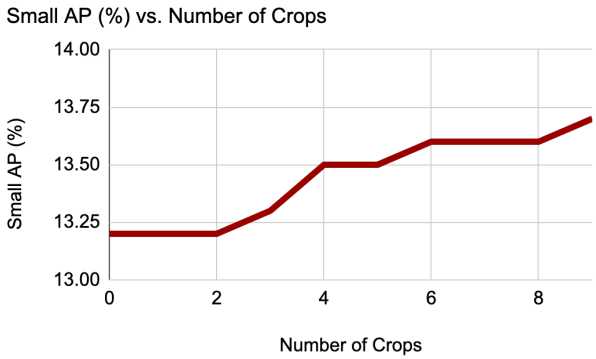


Figure 13: Effect of number of random crops on AP_S.

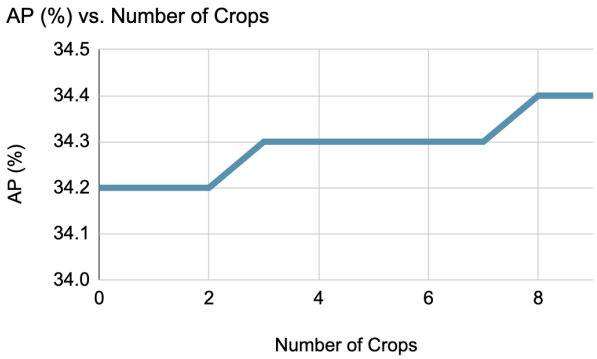


Figure 14: Effect of number of random crops on AP.

In both figures above, a crop count of 0 denotes the baseline. As the number of crops increase,

both AP and AP_S increases. The best performance is achieved when we used nine crops, which resulted in an AP of 34.4% (+0.2%) and AP_S of 13.7% (+0.5%). Although using random crops for score adjustment does yield positive results, it is computationally more expensive than using FiveCrops, which achieves the same improvement in AP as using nine randomm crops and +0.2% more improevement in AP_S. When only five random crops are used, AP and AP_S drops to 34.3% and 13.5%, respectively, both lower than the performance of FiveCrops prediction.

4.5. Statistical Significance

While we saw improvements in AP on our test set, the increase in AP is often small (+0.3% in the best case). Although this small rise in performance may be attributed to the fact that we lacked training examples and resources, it may be the case that the methods we developed are not statistically significant, and that the improvement was a result of some random chance. Therefore, we perform statistical significance tests by the bootstrapping approach proposed by [4, 20].

We obtain 200 bootstrap samples of 200 images with replacement, and obtain AP statics from our methods on each bootstrap. We then build a 95% confidence interval using the difference in AP from our methods and the baseline method, by taking the 5th to 95th percentile of these differences. We say that our methods are statistically significant if 0 is not included in these confidence intervals. Finally, we found both copy-pasting small and resized medium, large objects, as well as FiveCrops prediction using modified NMS, statistically significant.

5. Conclusion

The COCO dataset provides minimal information about small objects compared to their medium and large counterparts, and the lack of information contributes to the poor performance of state-of-the-art detectors on small objects. Therefore, to improve small object performance, we focus on supplying models with more examples of small objects at different stages of a model pipeline, either by copy-pasting downsampled medium and large objects to supply more small object examples for training preprocessing, or using upsampled small objects to increase their confidence scores for postprocessing during inference.

Going forward with the project, there're still much room for future research. First, we used a very small training set and test set, and it would be interesting to see if the techniques presented here would produce better results when the models are trained from scratch on the entire COCO train set with more than 100K images, given the computational resources. Second, we should test if the performance improvements we obtained from EfficientDet-D0 are applicable to other variants of EfficientDet and other models, especially two stage detectors like Faster R-CNN [18], as well as instance segmentation tasks. Third, since our study is entirely based on COCO, we need to examine whether or not our techniques are generalizable to other object detection dataset like PASCAL VOC [5] and TinyPerson [22].

6. Acknowledgements

I would like to thank Professor Olga Russkovsky for her guidance on this project, and the School of Engineering and Sciences for providing funding for this work.

References

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.
[2] D. Bolya *et al.*, "Tide: A general toolbox for identifying object detection errors," 2020.
[3] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," 2017.
[4] M. Everingham *et al.*, "Assessing the significance of performance differences on the pascal voc challenges via bootstrapping," 2013.
[5] M. Everingham *et al.*, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, p. 303–338, jun 2010. Available: https://doi.org/10.1007/s11263-009-0275-4
[6] C.-Y. Fu *et al.*, "Dssd : Deconvolutional single shot detector," 2017.
[7] G. Ghiasi *et al.*, "Simple copy-paste is a strong data augmentation method for instance segmentation," 2021.
[8] R. Girshick *et al.*, "Detectron," https://github.com/facebookresearch/detectron, 2018.
[9] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *ECCV*, 2012.
[10] M. Kisantal *et al.*, "Augmentation for small object detection," 2019.
[11] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," 2019.
[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.*, "Microsoft coco: Common objects in context," 2015.
[16] S. Liu *et al.*, "Path aggregation network for instance segmentation," 2018.
[17] 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
[18] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.
[19] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.
[20] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*, ser. Springer Texts in Statistics. New York: Springer, 2004.
[21] R. Wightman. Available: https://github.com/rwightman/efficientdet-pytorch

[22] X. Yu *et al.*, "Scale match for tiny person detection," 2019.
[23] F. Zhang *et al.*, "Multiresolution attention extractor for small object detection," 2020.