Article

Improving the performance of object detection by preserving label distribution

Heewon Lee, Sangtae Ahn*

School of Electronic and Electrical Engineering, Kyungpook National University, 80 Daehak-ro, Buk-gu, Daegu, South Korea, 41566

* Correspondence: Sangtae Ahn (stahn@knu.ac.kr)

Abstract: Object detection is a task that performs position identification and label classification of objects in images or videos. The information obtained through this process plays an essential role in various tasks in the field of computer vision. In object detection, the data utilized for training and validation typically originate from public datasets that are well-balanced in terms of the number of objects ascribed to each class in an image. However, in real-world scenarios, handling datasets with much greater class imbalance, i.e., very different numbers of objects for each class , is much more common, and this imbalance may reduce the performance of object detection when predicting unseen test images. In our study, thus, we propose a method that evenly distributes the classes in an image for training and validation, solving the class imbalance problem in object detection. Our proposed method aims to maintain a uniform class distribution through multi-label stratification. We tested our proposed method not only on public datasets that typically exhibit balanced class distribution but also on custom datasets that may have imbalanced class distribution. We found that our proposed method was more effective on datasets containing severe imbalance and less data. Our findings indicate that the proposed method can be effectively used on datasets with substantially imbalanced class distribution.

Keywords: object detection; imbalanced class distribution;

Citation: Lee, H; Ahn, S Improving performance of object detection by preserving label distribution. *Preprints* **2023**, *1*, 0. https://doi.org/

1. Introduction

Computer vision is a field of artificial intelligence that enables machines to understand and interpret visual information [1]. Computer vision involves the recognition and extraction of patterns from images or videos, and it has been applied in various fields [2–17]. Examples of its usage include object detection [18], image classification [19], face recognition [20], or image generation [21]. Among these, object detection is an important task in the field of computer vision, which aims to identify and localize multiple objects in images or videos.

To develop a model for object detection, the first step is to collect data and then appropriately split them into training and validation datasets. When splitting the original data, maintaining a similar class distribution between the training and validation datasets is crucial because datasets for object detection are presented as multi-label problems in that multiple objects can be present in a single image. In particular, the number of objects for each label could be imbalanced (Figure 1). Maintaining a similar class balance when dividing the dataset could be difficult. For image classification problems, a strategy called stratification can be used to maintain similar class distributions. However, it is unknown how stratification is applied to an object detection problem and how stratification influences the perofmrance of object detection.

Here, we present a study that proposes a new method called stratification for object detection (SOD), which applies stratification to the object detection problem. This method uses a multi-label stratification technique to preprocess labeled data in object detection and then applies stratification. This method facilitates the preservation of the class distribution among the split datasets, yielding better performance on public and custom datasets. Our proposed method can solve the problem of class imbalance better than existing methods, improving the performance of trained models. Our method was applied to object detection

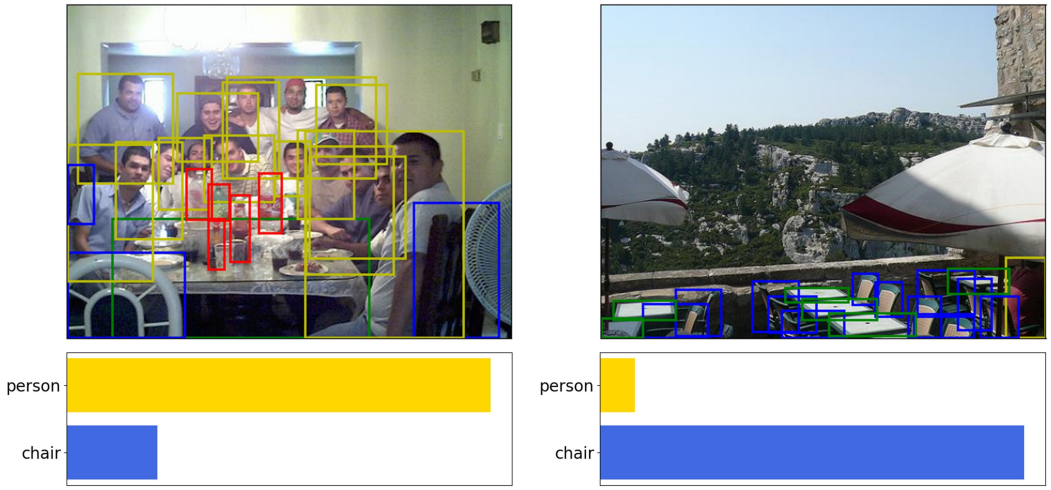


Figure 1. Each image has a different number of bounding boxes for each class. For example, the number of bounding boxes for a person may be high while the number of bounding boxes for a chair may be low (left) or vice versa (right). Images are taken from the Pascal Visual Object Classes(VOC) 2012 dataset.

problems, particularly implemented in the YOLO format because the YOLO algorithm is a widely used platform in object detection studies. Therefore, our proposed method is expected to be a useful tool that solves an important problem in the field of object detection and guarantees improved performance. Furthermore, it can contribute to improving the accuracy and reliability of the object detection model.

The main contributions of this research are as follows:

- We propose a method for preserving class distribution in object detection tasks.
- We experimentally demonstrate the effectiveness of our proposed method on both public datasets and custom datasets.

2. Related Work

2.1. Real-time object detection

Object detection is a computer vision task that involves detecting multiple objects in images or videos and classifying their positions and types. It is a key technological tool that enables computers to understand the real world and has been applied in various fields such as autonomous driving[22], surveillance systems[23], robotics[24], augmented reality[25], and medical image analysis[26]. Fundamentally, object recognition involves representing specific objects in an image as rectangular bounding boxes and characterizing them into different classes. While there are various algorithms for this task, we mainly employ the YOLO algorithm in this study.

YOLO[27], first proposed in 2015, is an algorithm that enables real-time object recognition. YOLO divides an image into a grid and applies a method to simultaneously predict bounding boxes and class probabilities for each grid cell. Unlike two-stage detectors[28], YOLO adopts a one-stage detector approach, whereby it considers the entire image at once and performs predictions. This unique feature of YOLO allows for real-time processing. Since 2015, YOLO has undergone many improvements. In YOLOv2[29], the first update of YOLO, the ability to detect objects of various sizes was enhanced. The concept of anchor boxes[30] was introduced in YOLOv2, and multi-scale training methods were used to train on images with different resolutions, resulting in improved detection of objects of different sizes. Additionally, YOLOv2 utilized the WordTree model to jointly train on the MSCOCO and ImageNet datasets, enabling the detection of over 9,000 different classes. YOLOv3[31] improved the detection of objects of various sizes by predicting bounding boxes in three different-sized feature maps. YOLOv3 also introduced a method to predict multi-labels for each box, facilitating the handling of more complex classification problems.

In YOLOv4[32], several optimizations were introduced to improve both performance and speed over those of previous versions. For YOLOv4, several features were introduced, namely, a new backbone network called CSPDarknet53, a new neck structure using PANet and SAM blocks, and the Mish activation function. These new structures and features made YOLOv4 more efficient and capable of accurately recognizing objects. The improved training speed, inference time, and model size were improved in YOLOv5[33], while user-friendly features were added using the PyTorch framework. This enabled faster and more efficient object recognition, expanding the practical application of object detection. Recently, the trainable bag-of-freebies method was introduced in YOLOv7[34,35] to significantly improve detection accuracy without increasing the inference cost. In this version, methods were also proposed to address issues arising from re-parameterized modules replacing original modules and apply dynamic label assignment strategies for different output layer assignments. Furthermore, extended and compound scaling methods were formulated to effectively utilize parameters and computations, reduce parameters and computational cost, and improve inference speed and accuracy. In this manner, YOLO-based models have continued to evolve and become the standard in real-time object detection. By examining these research trends, we can observe that object detection algorithms are steadily advancing to become faster, more accurate, and more applicable in diverse environments.

2.2. Stratification

Studies applying stratification to datasets have emphasized the significance of the class distribution of the dataset, which is particularly important in the treatment of classification problems in the field of machine learning[36]. Stratification is a type of data sampling technique that seeks to maintain the class distribution of the entire dataset in the training and validation sub-datasets. The key advantage of this technique is that it ensures that all classes are represented in the training and validation subsets as they are over the entire dataset, allowing the model to earn each class fairly.

The utilization of stratification in deep learning is mainly focused on addressing class imbalance issues[37]. In scenarios in which the number of samples for a specific class is significantly larger than that for other classes, i.e., class imbalance, the performance of the model can be excessively influenced by the dominant class. Stratification can alleviate such problems and ensure that all classes are fairly represented.

Furthermore, applying stratification to cross-validation is a highly effective strategy. Cross-validation[38] is a technique used to evaluate the generalization performance of a model by dividing the data into multiple folds and using each fold alternately as a training set and a validation set. By applying stratification to this process, the class distribution of the data in each fold can accurately reflect the distribution of the entire dataset. This can help verify whether each model performs equally well for all classes.

Various methods for applying stratification in deep learning have been utilized, and they can be adjusted depending on the characteristics of a specific problem and dataset. Stratification may be necessary in some cases but non-essential in others. However, in general, stratification is recognized as an important step for improving model training and enhancing generalization performance.

2.3. Multi-label stratification

Multi-label classification[39], unlike conventional classification, allows each data point to be classified with multiple labels simultaneously. Stratified sampling is a technique that selects samples evenly from each category, ensuring the representativeness of the entire dataset while maintaining the importance of each category. However, this technique is only applicable to single-label classification problems. Multi-label stratification extends this technique to problems in which each data point can have multiple labels. This ensures that each label combination is evenly distributed in the training and validation datasets, allowing the model to optimize generalizability for each label combination during training.

There are various use cases for multi-label stratification. For example, in music classification[40], as a song can belong to multiple genres, the corresponding training and validation datasets should represent each genre combination correctly to achieve accurate model learning. Multi-label stratification can be used to meet these requirements. Similarly, in medical imaging[41], an image can display multiple diseases. In this case, the combinations involving each disease label need to be well represented in the training and validation datasets, and multi-label stratification can be used for this purpose. Additionally, in text classification[42], text data such as news articles, research papers, and blog posts can be simultaneously classified multiple topics or categories. In this case, multi-label stratification can be applied to ensure that combinations involving each topic are evenly distributed in the training and validation datasets. In summary, multi-label stratification plays an important role in enabling more accurate model training in various fields to address real-world problems effectively.

3. Proposed algorithm

In this section, we introduce a new algorithm that applies stratification to object detection datasets using the YOLO format. The main goal is to improve existing dataset partitioning methods, enabling a more precise and fairer training process. The detailed operation of the algorithm is as follows.

This algorithm requires three inputs: the paths to the folders where the image files and text files are stored, denoted as F_{img} and F_{txt} respectively, and the number of subsets within the dataset to be created, denoted as k.

The algorithm, illustrated below, operates as follows. Lines 1 to 5 generate a list using the names of the image files. Similarly, lines 6 to 10 construct a list using the names of the text files. Lines 11 to 27 convert the label data from the text files into Dataframe format. In this process, if the dataset contains background images to prevent false positives, there may be no corresponding text file for that image or the text file may be empty to prevent the presence of false positives. To account for such background images in the partitioning process, we insert -1 in the class column of the Dataframe. Once this step is completed, the Dataframe contains the paths of the text files, number of objects per class, and the x and y coordinates, width (w), and height (h) of each object. Lines 29 to 34 are preprocessing steps needed before the multi-label stratified KFold procedure is performed. In this process, we perform one-hot encoding on the classes and concatenate the original Dataframe with the one-hot encoded Dataframe. We then multiply the w and h values by 1000 to prevent loss of values when calculating their averages. Next, we remove the "class", "x", and "y" columns from the Dataframe and group them based on filename. This creates a Dataframe that indicates how many objects of each class exist within each text file. For background images, the total object count is 0; thus, we change it to 1 to avoid division by 0 when calculating the averages of w and h. We then calculate w, h, and the ratio of h to w. Then, we remove the "w" and "h" columns to improve computational efficiency during the partitioning process. Finally, lines 35 to 38 perform the multi-label stratified KFold procedure, dividing the dataset into training and validation sets.

We discuss the selection of the labels for the Dataframe in the multi-label stratified KFold process $class_0 \sim class_n$, avg_w, avg_h, and avg_ratio. $class_0 \sim class_n$ indicates the presence and quantity of each class within an image. This parameter serves to ensure diversity in the training data by considering the various class labels within the dataset. We also selected average width (avg_w), average height (avg_h), and average ratio (avg_ratio) as labels for the following reason. Object detection models such as Faster R-CNN[43] and YOLO use predefined anchor boxes with specific aspect ratios. Anchor boxes are one of the elements used in object detection, in which frames with specific positions and sizes are set within an image. This allows the model to detect objects of various sizes and shapes, resulting in improved accuracy. If the aspect ratio distribution of bounding boxes in the training set and validation set differs, model performance can be negatively affected. If the aspect ratio distribution is not aligned, it becomes difficult to appropriately

Algorithm 1 YOLOstratifiedKFold

Input: F_{img}, F_{txt}, k

$L_{img} \leftarrow$ empty list ▷ List of image file names
for f in $listdir(F_{img})$ with $.jpg$ extension **do**
    append $splitext(f)_0$ to L_{img}
end for
$L_{img} \leftarrow$ remove duplicates from L_{img}

$L_{txt} \leftarrow$ empty list ▷ List of txt file names
for f in $listdir(F_{txt})$ with $.txt$ extension **do**
    append $splitext(f)_0$ to L_{txt}
end for
$L_{txt} \leftarrow$ remove duplicates from L_{txt}

$L_{data} \leftarrow$ empty list ▷ List of files data
for each F_{name} in L_{img} **do**
    if F_{name} is not in L_{txt} **then**
        $L_{data} \leftarrow$ append $[F_{name} + '.jpg', -1, None, None, None, None]$
    else
        $txt\_file\_path \leftarrow$ join F_{txt}, $F_{name} + '.txt'$
        $f \leftarrow$ open txt_file_path
        $lines \leftarrow$ read all lines from f
        if $lines$ is not empty **then**
            $L_{data} \leftarrow$ append $[F_{name} + '.jpg', -1, None, None, None, None]$
        else
            for each $line$ in $lines$ **do**
                $L_{data} \leftarrow$ append $[F_{name} + '.jpg', line_0, line_1, line_2, line_3, line_4]$
            end for
        end if
    end if
end for

$data \leftarrow$ create Dataframe from L_{data}

$one\_hot \leftarrow$ convert $data['class']$ to one-hot encoding
$data \leftarrow$ concatenate $data$ and one_hot and multiply 1000 to 'w', 'h' columns
$new\_df \leftarrow$ drop 'class', 'x', 'y' columns from $data$ and group by 'filename' and sum
$new\_df_{cnt} \leftarrow$ replace 0 with 1 in count of box
$new\_df_{avg\_w}, new\_df_{avg\_h}, new\_df_{avg\_ratio} \leftarrow$ Calculate average width, height, ratio
drop 'w','h' columns from new_df

$mskf \leftarrow$ initialize **MultilabelStratifiedKFold** with k

for each $(train\_idx, val\_idx)$ in $mskf.split(new\_df_{filename}, new\_df.iloc[:, 1:])$ **do**
    $X\_train, X\_val \leftarrow$ select rows from new_df by $train\_idx, val\_idx$
end for

should we discretize after this step

set the position and size of the anchor boxes, which ultimately affects the accuracy of object detection. Therefore, by selecting the average width (avg_w), average height (avg_h), and average ratio (avg_ratio) as labels, we aim to characterize the aspect ratio distribution of the bounding boxes in each dataset and optimize their size and aspect ratios, improving model performance. This allows the model to effectively detect objects with various aspect ratios.

4. Experiments

4.1. Datasets

We present the results of experiments conducted on public datasets to demonstrate the efficiency of the proposed algorithm. The datasets are used for object detection purposes, with each image indicating the location and type of object through a bounding box. Across the datasets, the number of samples is generally far greater than the number of classes, which enhances the training process by providing a wide range of samples for each class. In addition, we selected a dataset that includes background images to minimize the phenomenon of background false positives. This is crucial in preventing errors in object detection owing to features in the background.

We used entropy (Equation 1) to assess the distribution of each dataset. Higher entropy corresponds to higher uncertainty in the data and vice versa. Thus, entropy can be used as a measure of how well the classes in the data are distributed. In the equation, p_i represents the occurrence probability of each class over the entire dataset.

$$Entropy = -\sum_{i=1}^{n} p_i \log(p_i) \tag{1}$$

Table 1 shows a detailed analysis of the datasets. In datasets with a large number of classes, entropy tends to be relatively high. If the class label distribution of a dataset is diverse or uncertain, the stratification process may not be as effective. As stratification involves extracting samples while maintaining the class ratio of the original dataset, the proposed algorithm is recommended for use in scenarios in which the number of classes is less than or equal to 20.

Table 1. Statistical analysis of datasets for object detection. Public datasets: COCOval2017, Pascal VOC 2012 val, and PlantDoc. Private datasets[47]: Website screenshot, Aquarium, and BCCD.

Category	Dataset	Classes	Samples	Samples per Class	Samples per Image			Class per Image			Entropy
					Min	Avg	Max	Min	Avg	Max	
Public	COCO val2017[44]	80	5000	62.5	0	7.9	96	0	2.9	14	3.39
	Pascal VOC 2012 val[45]	20	3422	171.1	0	2.3	23	0	1.4	5	2.31
	PlantDoc[46]	30	2569	85.6	0	3.4	42	0	1	3	3.17
Private	Website screenshot	8	1206	150.8	2	45	2023	2	5.3	8	1.61
	Aquarium	7	638	91.1	0	7.6	56	0	1.4	3	1.42
	BCCD	3	364	121.3	1	13.4	30	1	2.5	3	0.53

4.2. Class distribution

We applied 10-fold cross-validation to the datasets and compared the class distribution of the original dataset with that of the split datasets. To quantify the differences in distribution between them, we used the mean absolute error(MAE) (Equation 2). This calculation was used to measure the difference in class ratios between the original dataset and split dataset. Through this, we quantitatively evaluated how accurately the split dataset reflects the class distribution of the original dataset. In the MAE metric below, y_i represents the class ratio of the original dataset, $\hat{y}_i$ represents the class ratio of the divided dataset, and n represents the number of classes in the existing dataset.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2}$$

We compared the MAE of the traditional KFold cross-validation method and that of our proposed algorithm. As shown in Table 2, in the majority of datasets in which our proposed algorithm was applied, the median MAE was lower compared to that of the traditional KFold cross-validation. This demonstrates that our proposed algorithm was

more effective in preserving the class ratios than the traditional method. However, some of datasets with an entropy of 2 or higher have found that KFold preserves class distribution better than our proposed algorithm. This indicates that such a phenomenon occurs when there is high uncertainty in the label distribution. In contrast, for datasets with an entropy of 2 or lower, our algorithm consistently showed lower median MAE and variance values compared to KFold in all cases. These results demonstrate that our algorithm provides a more stable and consistent performance.

Based on the experimental results, our proposed algorithm has been confirmed to effectively preserve class ratios while reducing variance, surpassing the commonly used KFold method.

Table 2. Statistical characteristics of the subsets generated through 10-fold cross-validation. A comparison between KFold and our proposed algorithm is presented, listing the name of each dataset, entropy of each dataset, class distribution of the training set (9 folds), and class distribution of the validation set (1 fold). The unit of the MAE is 1e-7.

Category	Dataset	Entropy	Train		Validation	
			KFold	Ours	KFold	Ours
Public	COCO val2017	3.39	**165±127**	168±128	**1466.5±1126.5**	1506.5±1144.5
	Pascal VOC 2012 val	2.31	**591.5±408.5**	618±391	5299±3712	**5279±3330**
	PlantDoc	3.17	463.5±321.5	**301.5±255.5**	4097±2803	**2614.5±2205.5**
Private	Website screenshot	1.61	4864±3880	**4092.5±3428.5**	42897±34009	**35538.5±29582.5**
	Aquarium	1.42	5172.5±4075.5	**3943±2202**	44031.5±33452.5	**38541±22795**
	BCCD	0.53	1973.5±1417.5	**1224±862**	17683.5±12738.5	**11459±8186**

4.3. Training and testing

To analyze the difference in performance between KFold and our proposed algorithm, we applied 10-fold cross-validation using the two methods in question to split the dataset. The dataset used in our study was divided according to the following procedure (Figure 2).

1. The original dataset was split into 10 folds using our proposed algorithm.
2. The last fold (10th fold) was set as the test dataset.
3. The remaining 9 folds (k-1 folds) were combined and further split into 9 folds using KFold.
4. Similarly, the k-1 folds were split into 9 folds using our proposed algorithm.
5. Training was performed iteratively for each of the 9 fold datasets.
6. The trained models were used for inference on the fixed test dataset.
7. Finally, the inference results using KFold and our proposed algorithm were compared using MAE.

The training was conducted with a 320-pixel image as the base model of YOLOv7 on a computing system consisting of three NVIDIA RTX 3090 Ti GPUs.

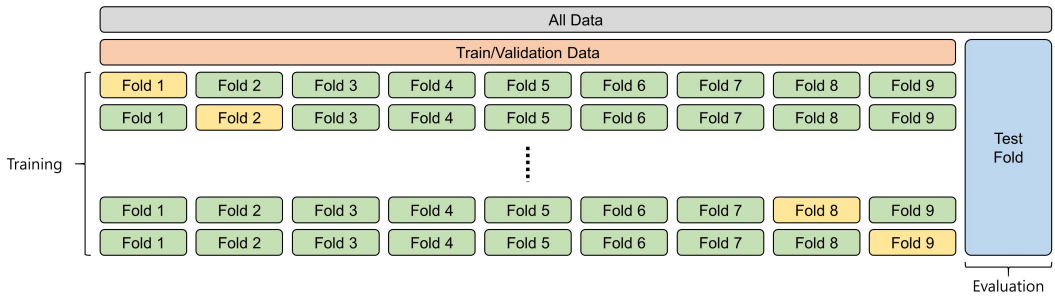


Figure 2. 10-fold cross-validation split method. The yellow box represents the fold used for validation, and the green boxes represent the fold used for training. When splitting the Train/Validation data and Test data, YOLOstratifiedKFold is used, and when dividing the data onto folds, either KFold or YOLOstratifiedKFold is used.

For datasets in which the entropy value is 2 or less, the training on the dataset applied with our proposed method consistently showed higher performance in the $mAP\_0.5$ metric as compared to when the KFold method was used, illustrated in Table 3. In the custom datasets, which contain relatively few classes, the class distribution of the datasets applied with our proposed method was more similar to the original class distribution (refer to Table 2) than the public dataset, and we confirmed that the performance of our model was also higher than that of the original KFold method. These results suggest that in training with data that exhibit relatively low complexity, i.e., a low number of classes, our method can achieve higher performance based on the $mAP\_0.5$ metric.

Table 3. Results of training by 10-fold cross validation

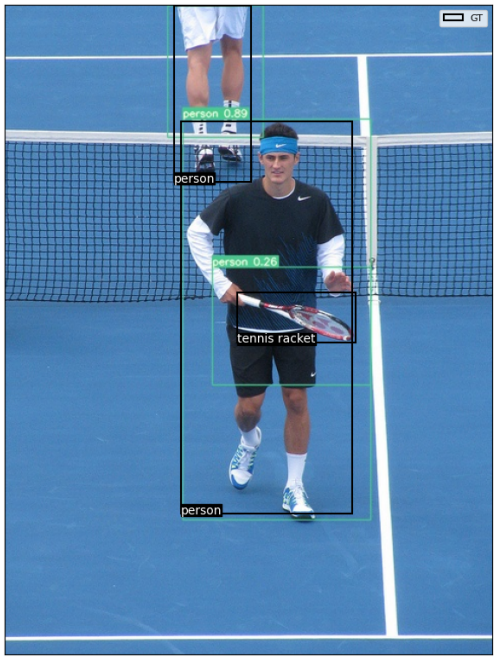
Category	Dataset	Entropy	mAP_0.5		mAP_0.5:0.95	
			KFold	Ours	KFold	Ours
Public	COCO val2017	3.39	23.50%±0.80%	**23.75%±1.25%**	14.00%±0.80%	**14.10%±0.70%**
	Pascal VOC 2012 val	2.31	**46.05%±2.25%**	44.95%±2.85%	**27.95%±1.05%**	27.45%±1.55%
	PlantDoc	3.17	**48.80%±2.60%**	48.20%±2.00%	32.90%±1.90%	**33.30%±1.30%**
Private	Website screenshot	1.61	45.85%±1.65%	**46.10%±1.50%**	**28.00%±1.20%**	27.75%±1.45%
	Aquarium	1.42	51.60%±3.20%	**52.25%±4.25%**	23.10%±2.10%	**23.25%±2.25%**
	BCCD	0.53	87.90%±1.70%	**88.55%±1.35%**	**55.55%±1.25%**	55.45%±1.55%

In contrast, it was difficult to confirm a significant improvement in the $mAP\_0.5:0.95$ metric from the training results. For our method, the datasets demonstrating better performance in $mAP\_0.5:0.95$ had less than 100 samples per class. These results suggest that our method effectively works even with small datasets, demonstrating good performance even for low numbers of samples per class. This implies that when the dataset is split using our method, the model can recognize the rough location of the object more accurately; however, it does not reflect the detailed shape and exact location of the object during the splitting process. One possible way to solve this problem is to perform stratification by referring to the features within the bounding box in the image; however, this method significantly increases the computational load required for stratification, which could greatly increase the amount of time to perform the routine. Therefore, in this study, we minimized the time burden and verified the performance of the model by excluding image references and only using label data stored in text files.

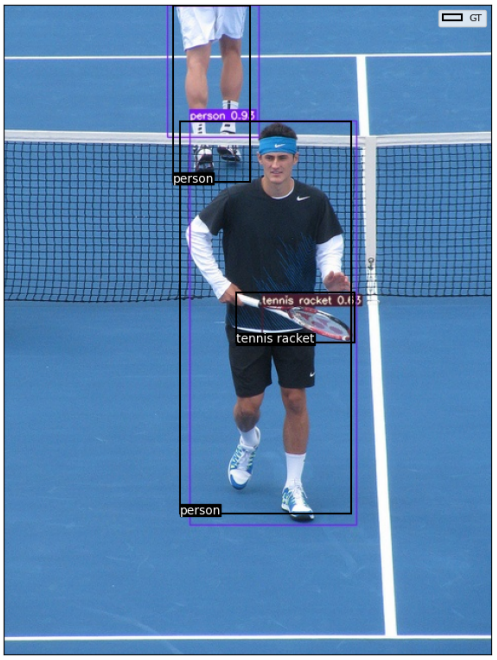
4.4. Inference results

In Figure 3, the inference results for the public dataset are presented. When comparing KFold and our method on the COCO dataset, the classification error for our method is less than that using KFold. While the tennis racket located in the center of the image is not accurately classified with KFold, it is correctly classified using our method. This demonstrates that using our method to split the dataset results in a smaller classification error as the class ratio is maintained throughout the dataset partitioning. A comparison conducted on the Pascal VOC 2012 dataset shows that the bounding box appearing in the upper left of the image is drawn incorrectly when using KFold but is drawn closer to the correct answer when using our method. This is also due to the reduction in localization error[48] as the aspect ratio of the bounding box is maintained with the split of the dataset. For the PlantDoc dataset, using KFold resulted in two bounding boxes with different classes drawn in a similar location even though there is only one ground truth, leading to multiple detections. This occurs because the class ratios between the subsets are not maintained in the random splitting of the dataset with this method. Conversely, this phenomenon did not appear using our method.

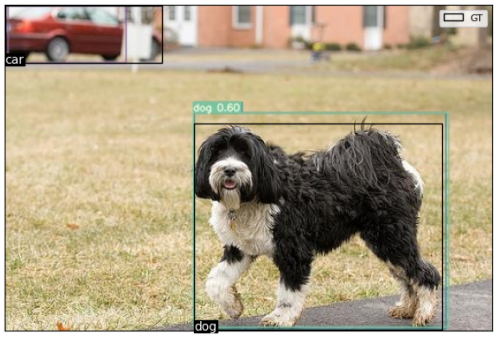
In Figure 4, the inference results on the private dataset are shown. In the Website screenshot dataset, there is a True Negative[49] owing to the complexity of features within a class, but our method shows a higher IoU than does KFold. In this case, the localization error is reduced by maintaining the aspect ratio of the bounding box similar as before after splitting the dataset. In the Aquarium dataset, the True Negative phenomenon of



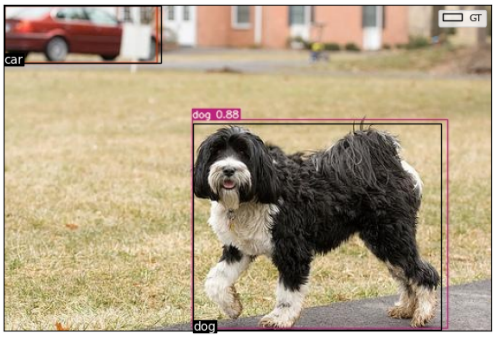
(a) (COCO val2017) KFold



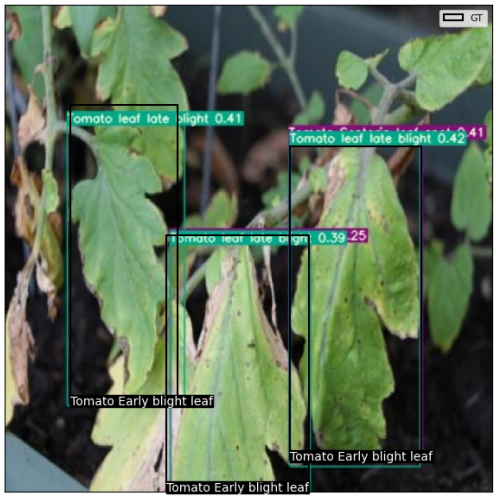
(b) (COCO val2017) Our method



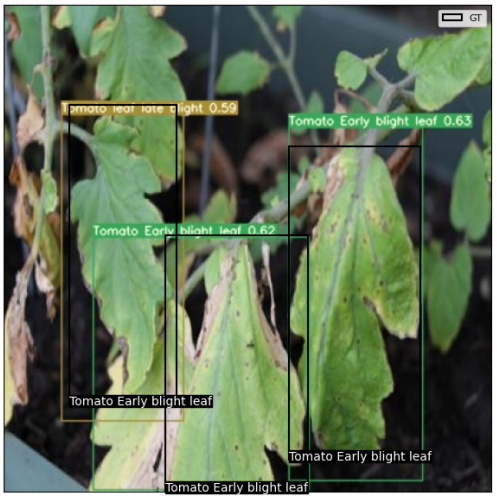
(c) (Pascal VOC 2012) KFold



(d) (Pascal VOC 2012) Our method

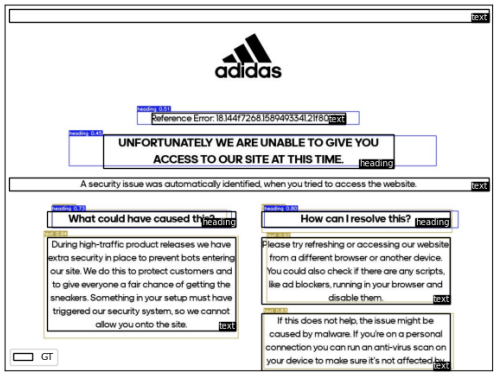


(e) (Plant) KFold

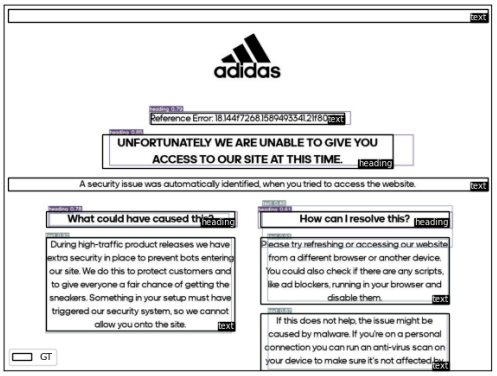


(f) (Plant) Our method

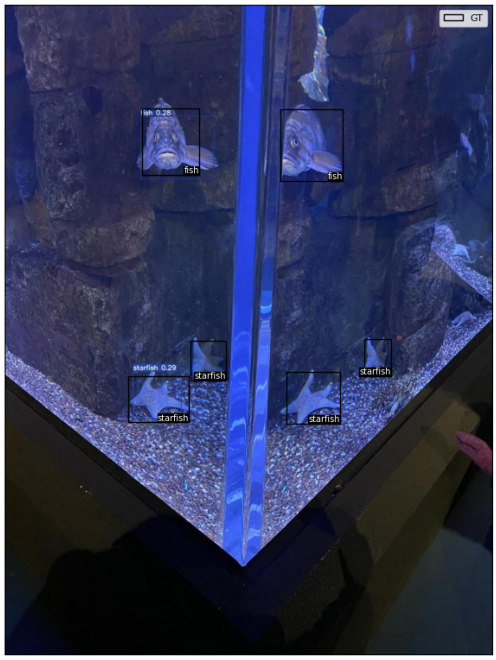
Figure 3. Comparing KFold to our proposed method on the Public dataset.



(a) (Website Screenshot) KFold

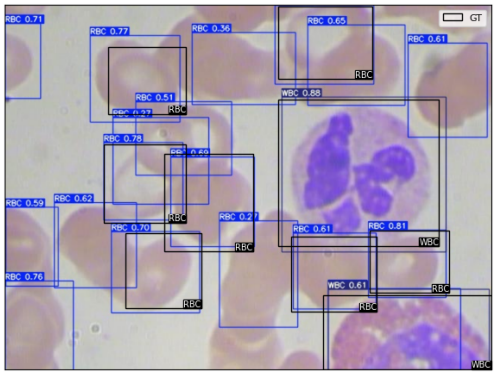


(b) (Website Screenshot) Our method

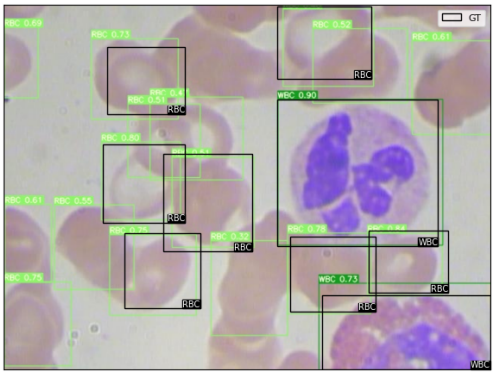


(c) (Aquarium) KFold

(d) (Aquarium) Our method



(e) (BCCD) KFold



(f) (BCCD) Our method

Figure 4. Comparing KFold to our proposed method on the private dataset.

no detection occurred when using KFold; however, this phenomenon was reduced when using our method. In the BCCD dataset, by optimizing the anchor box considering the class ratio, width, height, and ratio of the object, both KFold and our method achieved results closer to the ground truth of the bounding box and realized a higher IoU.

5. Discussion & Conclusion

This study focuses on the importance of applying stratification in the preprocessing stage of object detection tasks. This is the first study, to our knowledge, in the field of object detection that proposes a stratification method in dataset splitting, demonstrating the effect of alleviating class imbalance in the generation of training and validation sets. Additionally, stratification has been demonstrated to enhance the performance of an object detection model for a 2D image object detection dataset. However, stratification did not yield the best results under all experimental conditions. For high-dimensional datasets with a large number of classes, the application of stratification was ineffective, indicating that it may not be suitable for all types of datasets. Nevertheless, research involving classification tasks has shown that class balancing is essential and can significantly contribute to performance improvement for unbalanced datasets. This aspect is applicable to the object detection task in this study, emphasizing the need for an integrated approach to apply stratification in classification and object detection tasks. The results of this study confirm that the application of stratification must be carefully performed based on the characteristics of the dataset and distribution of the detection targets. As the features present in image data can influence the effect of stratification, further research in needed to improve and optimize the stratification method for use on images. Based on the results of this study, greater attention toward stratification utilization and its role in enhancing performance of object detection models is expected.

Author Contributions: Conceptualization, H.L., S.A.; methodology, H.L.; software, H.L.; validation, H.L., S.A.; formal analysis, H.L.; data curation, H.-L.; writing—original draft preparation, H.L.; writing—review and editing, H.L., S.A.; visualization, H.L.; funding acquisition, S.A. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Datasets used in this study are publicly available on the web(https://public.roboflow.com/).

Acknowledgments: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. NRF-2022R1A4A1023248)

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Voulodimos, Athanasios, et al. "Deep learning for computer vision: A brief review." Computational intelligence and neuroscience 2018 (2018).
2. Li, Zhuoling, et al. "CLU-CNNs: Object detection for medical images." Neurocomputing 350 (2019): 53-59.
3. Mao, Qi-Chao, et al. "Finding every car: a traffic surveillance multi-scale vehicle object detection method." Applied Intelligence 50 (2020): 3125-3136.
4. Yin, Jiale, et al. "The infrared moving object detection and security detection related algorithms based on W4 and frame difference." Infrared Physics & Technology 77 (2016): 302-315.
5. Cheng, Wen-Huang, et al. "Fashion meets computer vision: A survey." ACM Computing Surveys (CSUR) 54.4 (2021): 1-41.
6. Zhao, Jiawei, Rahat Masood, and Suranga Seneviratne. "A review of computer vision methods in network security." IEEE Communications Surveys & Tutorials 23.3 (2021): 1838-1878.
7. Sood, Shivani, and Harjeet Singh. "Computer vision and machine learning based approaches for food security: A review." Multimedia Tools and Applications 80.18 (2021): 27973-27999.
8. Lu, Yuzhen, and Sierra Young. "A survey of public datasets for computer vision tasks in precision agriculture." Computers and Electronics in Agriculture 178 (2020): 105760.
9. Tian, Hongkun, et al. "Computer vision technology in agricultural automation—A review." Information Processing in Agriculture 7.1 (2020): 1-19.

10. Scime, Luke, and Jack Beuth. "Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm." Additive Manufacturing 19 (2018): 114-126.
11. Qiu, Weichao, and Alan Yuille. "Unrealcv: Connecting computer vision to unreal engine." Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14. Springer International Publishing, 2016.
12. Xu, Yanling, et al. "Computer vision technology for seam tracking in robotic GTAW and GMAW." Robotics and computer-integrated manufacturing 32 (2015): 25-36.
13. Kazemian, Ali, et al. "Computer vision for real-time extrusion quality monitoring and control in robotic construction." Automation in Construction 101 (2019): 92-98.
14. Demir, Ilke, et al. "Deepglobe 2018: A challenge to parse the earth through satellite images." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.
15. Rad, Mohammad Saeed, et al. "A computer vision system to localize and classify wastes on the streets." Computer Vision Systems: 11th International Conference, ICVS 2017, Shenzhen, China, July 10-13, 2017, Revised Selected Papers 11. Springer International Publishing, 2017.
16. Grys, Ben T., et al. "Machine learning and computer vision approaches for phenotypic profiling." Journal of Cell Biology 216.1 (2017): 65-71.
17. Kremer, Jan, et al. "Big universe, big data: Machine learning and image analysis for astronomy." IEEE Intelligent Systems 32.2 (2017): 16-22.
18. Zou, Zhengxia, et al. "Object detection in 20 years: A survey." Proceedings of the IEEE (2023).
19. Rawat, Waseem, and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review." Neural computation 29.9 (2017): 2352-2449.
20. Kortli, Yassin, et al. "Face recognition systems: A survey." Sensors 20.2 (2020): 342.
21. Elasri, Mohamed, et al. "Image generation: A review." Neural Processing Letters 54.5 (2022): 4609-4646.
22. Feng, Di, et al. "A review and comparative study on probabilistic object detection in autonomous driving." IEEE Transactions on Intelligent Transportation Systems 23.8 (2021): 9961-9980.
23. Elhoseny, Mohamed. "Multi-object detection and tracking (MODT) machine learning model for real-time video surveillance systems." Circuits, Systems, and Signal Processing 39 (2020): 611-630.
24. Xu, Ge, et al. "The Object Detection, Perspective and Obstacles In Robotic: A Review." EAI Endorsed Transactions on AI and Robotics 1.1 (2022).
25. Liu, Luyang, Hongyu Li, and Marco Gruteser. "Edge assisted real-time object detection for mobile augmented reality." The 25th annual international conference on mobile computing and networking. 2019.
26. Li, Zhuoling, et al. "CLU-CNNs: Object detection for medical images." Neurocomputing 350 (2019): 53-59.
27. Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
28. Du, Lixuan, Rongyu Zhang, and Xiaotian Wang. "Overview of two-stage object detection algorithms." Journal of Physics: Conference Series. Vol. 1544. No. 1. IOP Publishing, 2020.
29. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
30. Zhong, Yuanyi, et al. "Anchor box optimization for object detection." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020.
31. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
32. Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
33. Jocher, Glenn, et al. "ultralytics/yolov5: v3. 0." Zenodo (2020).
34. Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.
35. Zhang, Zhi, et al. "Bag of freebies for training object detection neural networks." arXiv preprint arXiv:1902.04103 (2019).
36. Prusty, Sashikanta, Srikanta Patnaik, and Sujit Kumar Dash. "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer." Frontiers in Nanotechnology 4 (2022): 972421.
37. Wu, Qingyao, et al. "ForesTexter: An efficient random forest algorithm for imbalanced text categorization." Knowledge-Based Systems 67 (2014): 105-116.
38. Arlot, Sylvain, and Alain Celisse. "A survey of cross-validation procedures for model selection." (2010): 40-79.
39. Tsoumakas, Grigorios, and Ioannis Katakis. "Multi-label classification: An overview." International Journal of Data Warehousing and Mining (IJDWM) 3.3 (2007): 1-13.
40. Trohidis, Konstantinos, et al. "Multi-label classification of music into emotions." ISMIR. Vol. 8. 2008.
41. Nigam, Priyanka. Applying deep learning to ICD-9 multi-label classification from medical records. Technical report, Stanford University, 2016.
42. Yang, Bishan, et al. "Effective multi-label active learning for text classification." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009.

43. Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).
44. Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer International Publishing, 2014.
45. Everingham, Mark, et al. "The pascal visual object classes challenge: A retrospective." International journal of computer vision 111 (2015): 98-136.
46. Singh, Davinder, et al. "PlantDoc: A dataset for visual plant disease detection." Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. 2020. 249-253.
47. Ciaglia, Floriana, et al. "Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark." arXiv preprint arXiv:2211.13523 (2022).
48. Hoiem, Derek, Yodsawalai Chodpathumwan, and Qieyun Dai. "Diagnosing error in object detectors." European conference on computer vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
49. Padilla, Rafael, Sergio L. Netto, and Eduardo AB Da Silva. "A survey on performance metrics for object-detection algorithms." 2020 international conference on systems, signals and image processing (IWSSIP). IEEE, 2020.