mathematics

MDPI

Article

Improving the Performance of Object Detection by Preserving Balanced Class Distribution

Heewon Lee [1] and Sangtae Ahn [1,2,*]

[1] School of Electronics Engineering, Kyungpook National University, Daegu 41566, Republic of Korea; saebuk2000@knu.ac.kr
[2] School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
* Correspondence: 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 private 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: computer vision; object detection; imbalanced class distribution; multi-label stratification

MSC: 68T07

Citation: Lee, H.; Ahn, S. Improving the Performance of Object Detection by Preserving Balanced Class Distribution. *Mathematics* **2023**, *11*, 4460. https://doi.org/10.3390/math11214460

Academic Editor: Konstantin Kozlov

Received: 30 August 2023
Revised: 28 September 2023
Accepted: 26 October 2023
Published: 27 October 2023

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]. Examples of its usage include object detection, image classification, face recognition, and image generation. 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 [3], the initial step involves collecting data, which should subsequently be partitioned 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. Specifically, if the number of objects for each class is imbalanced, as depicted in Figure 1, maintaining a balanced class distribution during dataset partitioning may prove challenging. For image classification problems, a strategy called stratification can be used to maintain similar class distributions. To the best of our knowledge, no methods for applying stratification in the partitioning of datasets for object detection tasks have been previously reported; only random partitioning methods have been employed.

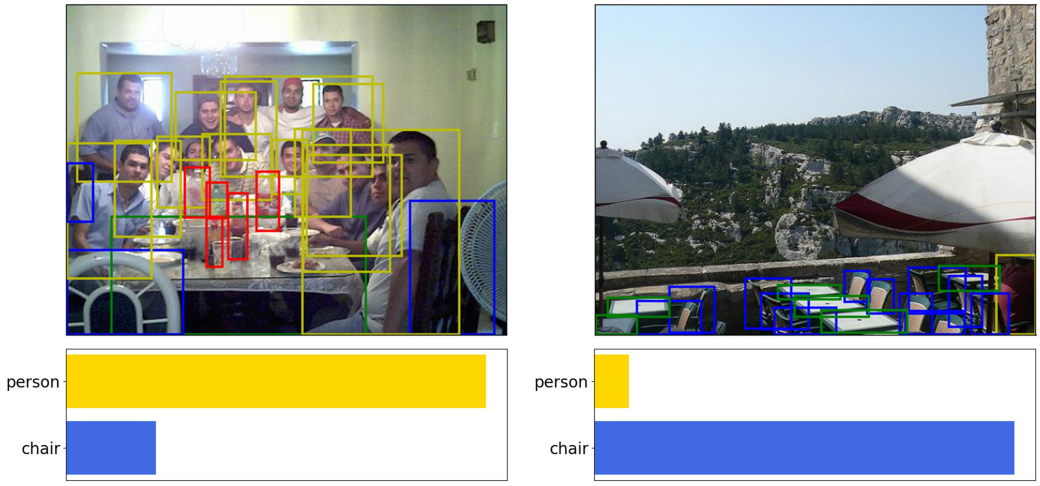


Figure 1. Each image has a different number of bounding boxes for each class. For example, the number of bounding boxes (yellow) for a person may be high, while the number of bounding boxes (blue) for a chair may be low (**left**), or vice versa (**right**). Images are taken from the Pascal Visual Object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

Here, we present a study that proposes a method for stratification for object detection (SOD) to improve the performance by balancing the class distribution. 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 private datasets. Our proposed method can solve the problem of class imbalance better than conventional methods, enhancing the efficacy of the trained models. Our method was applied to object detection problems, particularly implemented in the format of YOLO [4] because the YOLO algorithm is widely used in object detection studies. Therefore, our proposed method is anticipated to provide a significant contribution to addressing a critical challenge in the field of object detection and guarantees improved performance in scenarios involving class imbalances.

The main contributions of this research are as follows:

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

2. Related Works

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 [5], surveillance systems [6], robotics [7], augmented reality [8], and medical image analysis [9]. Fundamentally, object detection 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 introduce the YOLO algorithm in our study.

YOLO [4], 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 [10], 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 [11], the ability

to detect objects of various sizes was enhanced. The concept of anchor boxes [12] 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 9000 different classes. YOLOv3 [13] 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 multiple labels for each box, facilitating the handling of more complex classification problems. In YOLOv4 [14], 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. Improvements were made to training speed, inference time, and model size in YOLOv5 [15]; additionally, user-friendly features were integrated through the PyTorch framework. This advancement facilitated more rapid and efficient object recognition, thereby broadening the range of practical applications for object detection. Recently, the trainable bag-of-freebies method was introduced in YOLOv7 [16] to significantly improve detection accuracy without increasing the inference cost. In YOLOv7, 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 method in real-time object detection. By exploring these research trends, we observe that object detection algorithms are steadily advancing to become faster, more accurate, and more applicable in diverse environments.

2.2. Class Imbalance Problems in Object Detection

In the field of object detection, various sampling methods are being actively researched to address the issue of class imbalance during the training process [17]. These methods can be broadly categorized into hard sampling, soft sampling, sampling-free, and generative methods.

Hard sampling methods address class imbalance by selecting a subset of positive and negative examples from a given set of labeled bounding boxes. These methods rely on heuristic methods, and each selected instance contributes uniformly to the loss function. Instances that are not selected are excluded from the current training iteration. Key methods in this category include random sampling, hard example mining, and limiting the search space. Soft sampling methods adjust the weights of samples based on their relative importance, allowing all samples to contribute to the training process. Notable approaches include focal loss, gradient harmonizing mechanism, and prime sample attention. Sampling-free methods detect objects without generating anchor boxes, proposals, or undergoing a sampling process. Notable approaches include adding an object branch, optimally tuning hyperparameters, and directly modeling the final performance metric to evaluate examples. Generative methods tackle the class imbalance issue through direct data generation. They often employ generative adversarial networks to synthesize data and create feature maps for generating more challenging examples. These methods are integrated into the training pipeline in an end-to-end manner.

Previous methods primarily focus on resolving the class imbalance issue during the training process. However, our approach is unique in that it addresses class imbalance at the stage of splitting the dataset into training and validation sets prior to training.

2.3. 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 [18]. 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 appropriately represent each class. The utilization of stratification in deep learning is mainly focused on addressing class imbalance issues [19]. In scenarios in which the number of samples for a specific class is significantly larger than that for other classes, that is, 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 [20] 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.

2.4. Multi-Label Stratification

Multi-label classification [21], 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 applications for multi-label stratification. For example, in music classification [22], as a song can belong to multiple genres, the corresponding training and validation datasets should represent each genre combination correctly to achieve accurate training. Multi-label stratification can be used to meet these requirements. Similarly, in medical imaging [23], 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 [24], text data from sources 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 conclusion, multi-label stratification plays an important role in enabling more accurate model training in various fields to address practical or domain-specific challenges effectively.

3. Proposed Algorithm

In this section, we present a new algorithm for implementing stratification to datasets for object detection tasks. The primary objective of this algorithm is to optimize the existing dataset partitioning mechanisms to improve the model's performance. In the initial stages, we identified objects within each image that were classified according to their respective

classes and aggregated their counts, representing the class of each bounding box with one-hot encoding. Additionally, we identified the necessity for stratification to address the potential performance degradation due to the aspect ratio imbalance of bounding boxes. To that end, we measure the width and height of each bounding box and then group them by image. From the grouped data, we compute the frequency of each class as well as the average width and height, to derive the average aspect ratio for each image. Based on this information, we proceed with multi-label stratification. The detailed operation of the algorithm is as follows.

This Algorithm 1 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, explained below as a pseudo-code, 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 cases where the dataset includes background images to mitigate false positives, the corresponding text file for such an image may either be absent or empty. 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, the number of objects per class, and the x and y coordinates, width (w), and height (h) of each object. Lines 29 to 34 consist of preprocessing steps required prior to the multi-label stratified k-fold 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 file name. 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 k-fold procedure, dividing the dataset into training and validation sets.

We discuss the selection of the labels for the DataFrame in the multi-label stratified k-fold 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 reasons. Object detection models such as Faster R-CNN [25] 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 capability enables the model to detect objects of varying sizes and shapes, thereby enhancing model accuracy. If there is a discrepancy in the aspect ratio distribution of bounding boxes between the training and validation sets, the model's detection accuracy can be adversely affected. If the aspect ratio distribution is not aligned, it becomes difficult to appropriately 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 to optimize their size and aspect ratios, improving model performance. This methodology enables the model to more effectively detect objects with a range of aspect ratios.

Algorithm 1 Multi-Label Stratification for Object Detection

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 **Multi-Label Stratified K-Fold** 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

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 selected dataset is highly imbalanced, with the number of samples significantly exceeding 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 mitigate the issue of false positives caused by background features. 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 probability of occurrence for each class within the 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 be less 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 [26]: 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 [27]	80	5000	62.5	0	7.9	96	0	2.9	14	3.39
	Pascal VOC 2012 val [28]	20	3422	171.1	0	2.3	23	0	1.4	5	2.31
	PlantDoc [29]	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 approach, we conducted a quantitative evaluation to assess the accuracy with which 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 k-fold 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 k-fold. This demonstrates that our proposed algorithm was more effective in preserving the class ratios than the traditional method. However, some datasets with an entropy of 2 or higher have found that the k-fold 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 k-fold 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 k-fold method.

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

Category	Dataset	Entropy	Train		Validation	
			K-Fold	Ours	K-Fold	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**	42,897 ± 34,009	**35,538.5** ± **29,582.5**
	Aquarium	1.42	5172.5 ± 4075.5	**3943** ± **2202**	44,031.5 ± 33,452.5	**38,541** ± **22,795**
	BCCD	0.53	1973.5 ± 1417.5	**1224** ± **862**	17,683.5 ± 12,738.5	**11,459** ± **8186**

4.3. Training and Testing

To analyze the difference in performance between the different methods and our proposed algorithm, we applied 10-fold cross-validation using several methods in question to split the dataset, k-fold and our method, one of the soft sampling methods, FocalLoss. 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 k-fold.
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 k-fold and our proposed algorithm were compared using MAE.

The training was conducted by 300 epoch with a 320-pixel image as the base model of YOLOv7 on a computing system consisting of three NVIDIA RTX 3090 Ti GPUs.

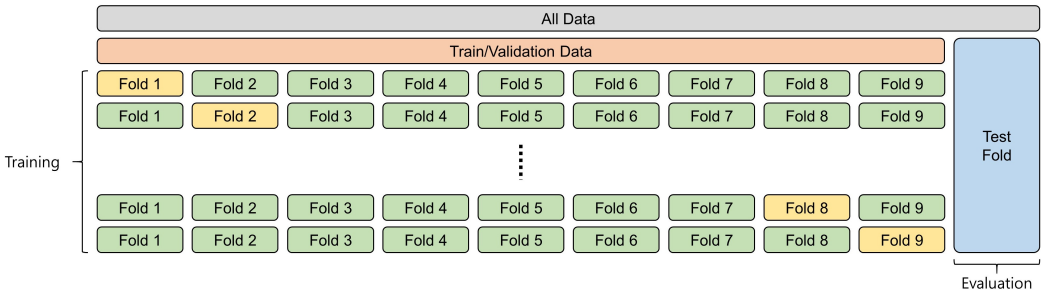


Figure 2. The 10-fold cross-validation split method for splitting the training, validation, and test data. The yellow box represents the fold used for validation, and the green boxes represent the fold used for training.

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 k-fold or FocalLoss method was used, as tabulated in Table 3. In the private 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 k-fold or FocalLoss 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.

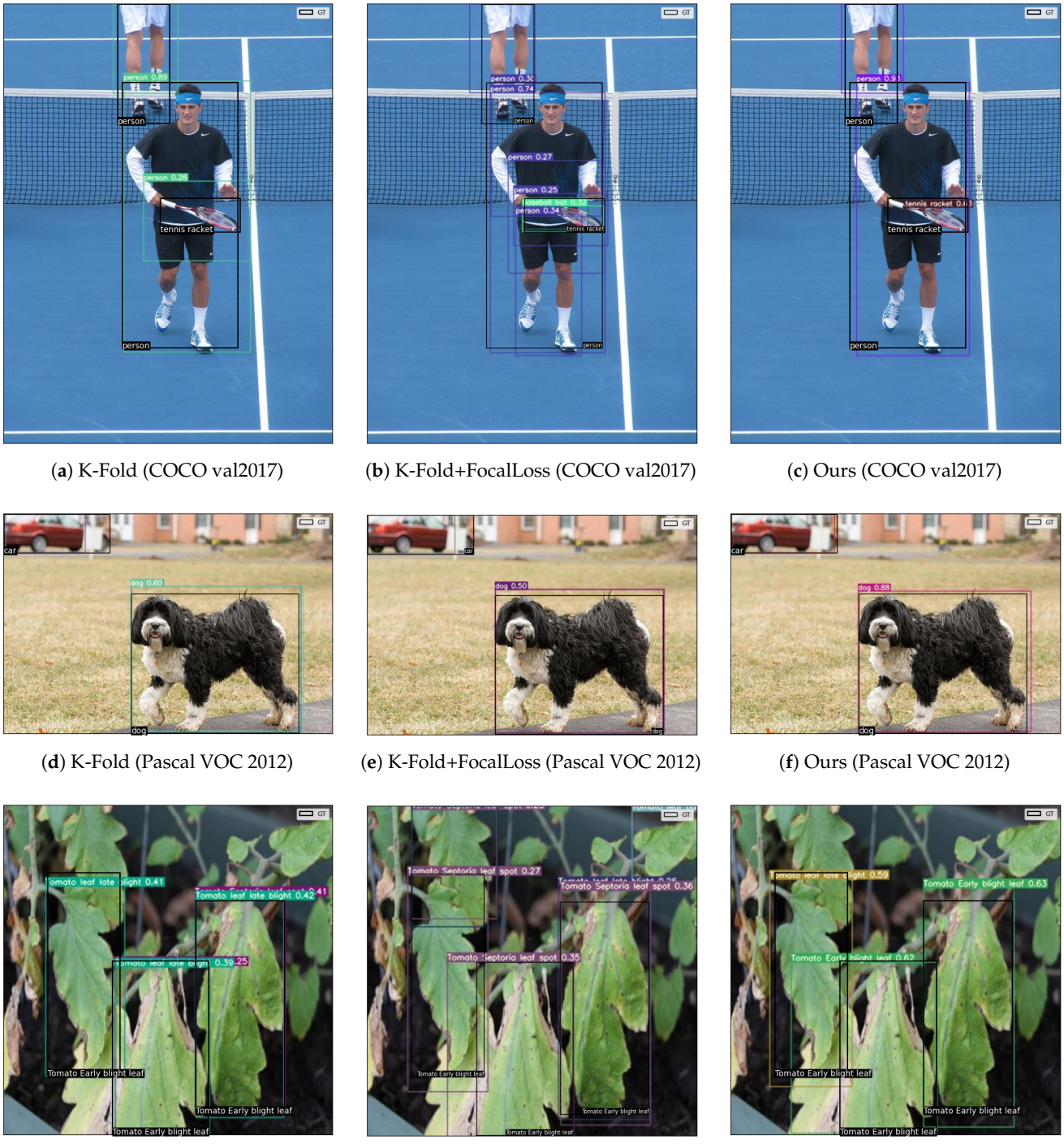
In contrast, it was difficult to confirm a significant improvement in the mAP_0.5:0.95 metric from the training results. For our method, the datasets demonstrating better performance in mAP_0.5:0.95 had less than 100 samples per class. These results suggest that our method effectively works even with 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.

Table 3. Comparison of classification performance in public and private datasets.

Category	Dataset	Entropy	Methods	mAP_0.5	mAP_0.5:0.95
Public	COCO val2017	3.39	K-Fold	23.50% ± 0.80%	14.00% ± 0.80%
			K-Fold + FL	16.80% ± 0.80%	10.46% ± 0.67%
			Ours	23.75% ± 1.25%	14.10% ± 0.70%
	Pascal VOC 2012 val	2.31	K-Fold	46.05% ± 2.25%	27.95% ± 1.05%
			K-Fold + FL	35.20% ± 1.40%	22.05% ± 1.15%
			Ours	44.95% ± 2.85%	27.45% ± 1.55%
	PlantDoc	3.17	K-Fold	48.80% ± 2.60%	32.90% ± 1.90%
			K-Fold + FL	21.20% ± 5.10%	14.80% ± 4.00%
			Ours	48.20% ± 2.00%	33.30% ± 1.30%
Private	Website screenshot	1.61	K-Fold	45.85% ± 1.65%	28.00% ± 1.20%
			K-Fold + FL	25.95% ± 1.15%	15.70% ± 0.60%
			Ours	46.10% ± 1.50%	27.75% ± 1.45%
	Aquarium	1.42	K-Fold	51.60% ± 3.20%	23.10% ± 2.10%
			K-Fold + FL	12.11% ± 3.09%	5.22% ± 1.44%
			Ours	52.25% ± 4.25%	23.25% ± 2.25%
	BCCD	0.53	K-Fold	87.90% ± 1.70%	55.55% ± 1.25%
			K-Fold + FL	79.85% ± 3.25%	51.35% ± 1.95%
			Ours	88.55% ± 1.35%	55.45% ± 1.55%

4.4. Comparison

In Figure 3, object detection results targeting public datasets are shown. When using the COCO dataset as a standard, a classification error occurred in the tennis racket located at the center of the image when applying the k-fold method. When both k-fold and FocalLoss were applied simultaneously, the classification error was reduced; however, the results indicated overconfidence in detection, deviating from real-world applicability. On the other hand, when our algorithm was applied, both the classification error and overconfidence significantly decreased compared to other methods.



(**a**) K-Fold (COCO val2017) (**b**) K-Fold+FocalLoss (COCO val2017) (**c**) Ours (COCO val2017)

(**d**) K-Fold (Pascal VOC 2012) (**e**) K-Fold+FocalLoss (Pascal VOC 2012) (**f**) Ours (Pascal VOC 2012)

(**g**) K-Fold (PlantDoc) (**h**) K-Fold+FocalLoss (PlantDoc) (**i**) Ours (PlantDoc)

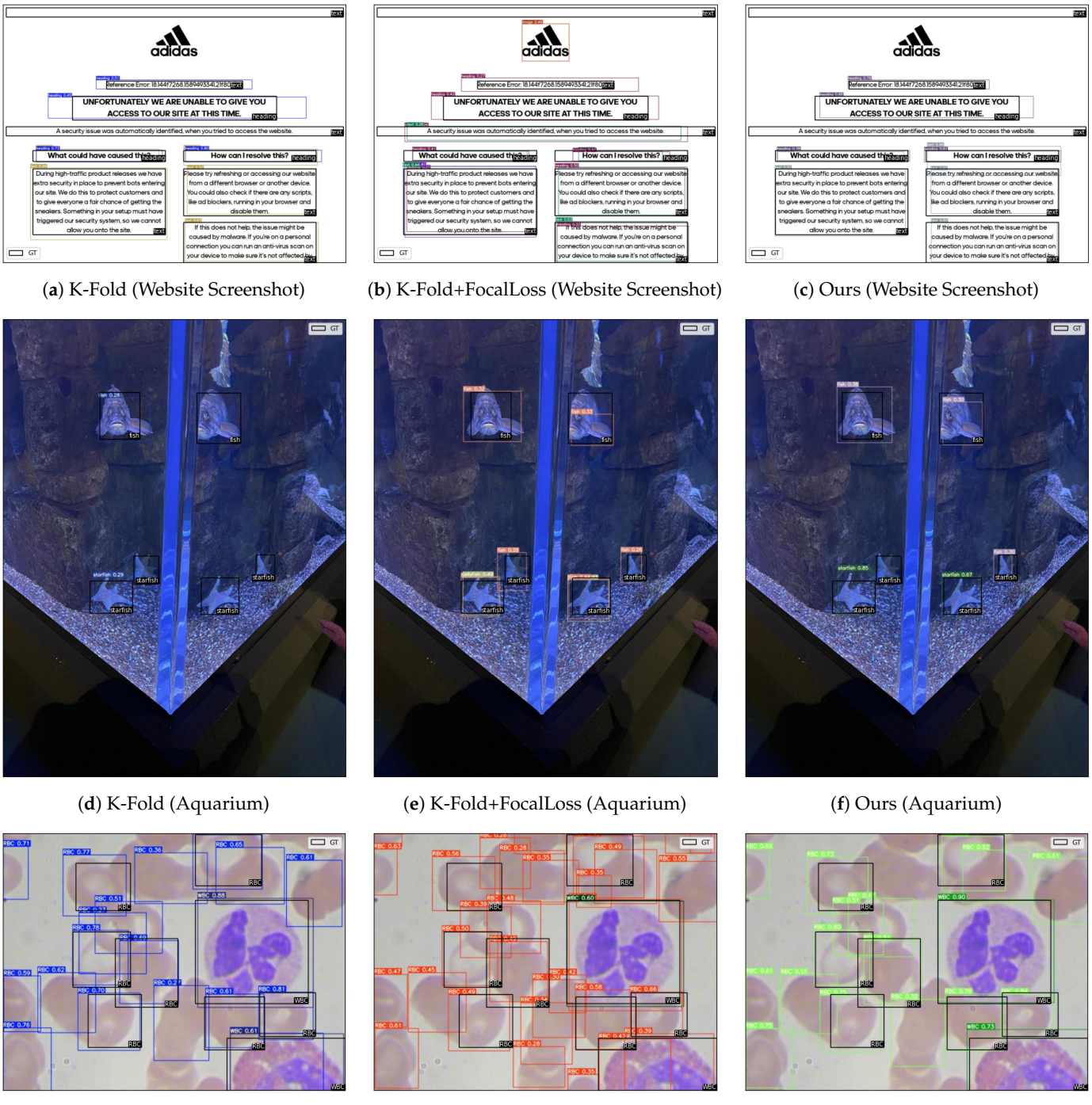
Figure 3. Comparison of object detection in k-fold and our method for the public datasets.

In the Pascal VOC 2012 dataset, when using the k-fold method, a bounding box regression error occurred with incorrect coordinates for the car located at the top-left corner of the image. When k-fold and FocalLoss were applied together, this error relatively decreased, but the model demonstrated underconfidence in the detection of the dog object. However, when we applied our method, the bounding box regression error was lower, and the issue of underconfidence was also mitigated.

In the PlantDoc dataset, when applying k-fold, a phenomenon of overlapping classes was observed. Additionally, issues related to underconfidence were also observed. When

both k-fold and FocalLoss were applied, these issues intensified. However, when our method was applied, both the phenomenon of overlapping classes and issues related to underconfidence were resolved.

In Figure 4, object detection results targeting private datasets are shown. When the k-fold method was applied to the website screenshot dataset, bounding box regression errors occurred. Using k-fold and FocalLoss together led to the existence of multi-class overlap; however, the bounding box regression errors were reduced compared to when only k-fold was used. When our methodology was applied, both bounding box regression errors and multi-class overlap were improved compared to other existing methods.



(**a**) K-Fold (Website Screenshot) (**b**) K-Fold+FocalLoss (Website Screenshot) (**c**) Ours (Website Screenshot)

(**d**) K-Fold (Aquarium) (**e**) K-Fold+FocalLoss (Aquarium) (**f**) Ours (Aquarium)

(**g**) K-Fold (BCCD) (**h**) K-Fold+FocalLoss (BCCD) (**i**) Ours (BCCD)

Figure 4. Comparison of object detection in k-fold and our method for the private datasets.

In the Aquarium dataset, true negatives were observed when k-fold was applied. When using k-fold and FocalLoss together, true negatives decreased, but bounding box regression errors and multi-class overlap phenomena appeared. When our method was applied, improvements were observed in true negatives, bounding box regression errors, and multi-class overlap compared to other methods.

In the BCCD dataset, when k-fold was applied, features of objects to be detected and background objects appeared similar, leading to background false positives (FPs). Using k-fold and FocalLoss together led to an increase in overconfidence, resulting in more background FPs. When our methodology was applied, overconfidence was reduced, and fewer background FPs were observed compared to other methods.

5. Discussion

Sampling methods that can be applied in the training process have been utilized. While the random splitting technique is often straightforward to implement, it can be problematic due to data imbalance and feature distribution, which can negatively impact the model's generalization capabilities. Traditional focal loss techniques prioritize difficult classes in an imbalanced dataset, but they can occasionally overcompensate, resulting in other types of errors. Our stratification technique effectively balances classes without introducing new sources of error, thus providing a more robust approach.

This study proposes a stratification technique that outperforms traditional methods such as random splitting, k-fold cross-validation in object detection tasks, and focal loss for addressing class imbalance during the loss computation phase. Additionally, this study suggests the importance of further research to improve stratification methods within the context of image data, where object features can significantly impact the effects of stratification. The anticipated increase in interest in the role of stratification for enhancing the performance of object detection models makes it important for future research to focus on the improvement and optimization of universal stratification techniques applicable to various types of object recognition datasets. By outlining these elements, this study serves as an important step in understanding the complexities and potential benefits of stratification in object detection tasks.

6. Conclusions

This study primarily aims at highlighting the benefits of applying stratification in object detection tasks. It stands as one of the first endeavors to introduce a stratification method in the dataset splitting phase, particularly focusing on mitigating class imbalance in the training and validation sets. The study has been successful in showing that the use of stratification enhances the performance of object detection models when tested on a 2D image object detection dataset. However, it is important to acknowledge the limitations of stratification as well. The technique did not yield favorable outcomes in all experimental conditions. Particularly, in high-dimensional datasets comprising a large number of classes, stratification was found to be ineffective. This suggests that the utility of stratification may vary depending on the characteristics of the dataset. Moreover, the study recognizes that class balancing is an essential aspect in classification tasks, which aligns well with the object detection tasks presented here. The results of this study emphasize the need for a holistic strategy that integrates both classification and object detection under the umbrella of stratification.

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

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2022R1A4A1023248 and No. RS-2023-00209794).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Datasets used in this study are publicly available on the web (accessed on 21 June 2023, https://public.roboflow.com).

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

References

1. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349 . [CrossRef] [PubMed]
2. Pathak, A.R.; Pandey, M.; Rautaray, S. Application of deep learning for object detection. *Procedia Comput. Sci.* **2018**, *132*, 1706–1717. [CrossRef]
3. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, 111, 257–276. [CrossRef]
4. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
5. Feng, D.; Harakeh, A.; Waslander, S.L.; Dietmayer, K. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 9961–9980. [CrossRef]
6. Elhoseny, M. Multi-object detection and tracking (MODT) machine learning model for real-time video surveillance systems. *Circuits Syst. Signal Process.* **2019**, *39*, 611–630. [CrossRef]
7. Xu, G.; Khan, A.S.; Moshayedi, A.J.; Zhang, X.; Shuxin, Y. The Object Detection, Perspective and Obstacles In Robotic: A Review. *EAI Endorsed Trans. AI Robot.* **2022**, *1*, e13. [CrossRef]
8. Liu, L.; Li, H.; Gruteser, M. Edge assisted real-time object detection for mobile augmented reality. In Proceedings of the 25th Annual International Conference on Mobile Computing and Networking, Los Cabos, Mexico, 21–25 October 2019.
9. Li, Z.; Dong, M.; Wen, S.; Hu, X.; Zhou, P.; Zeng, Z. CLU-CNNs: Object detection for medical images. *Neurocomputing* **2019**, *350*, 53–59. [CrossRef]
10. Du, L.; Zhang, R.; Wang, X. Overview of two-stage object detection algorithms. *J. Phys. Conf. Ser.* **2020**, *1544*, 012033. [CrossRef]
11. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
12. Zhong, Y.; Wang, J.; Peng, J.; Zhang, L. Anchor box optimization for object detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass Village, CO, USA, 1–5 March 2020.
13. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
14. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. YOLOv5 by Ultralytics (Version 7.0). Available online: https://github.com/ultralytics/yolov5 (accessed on 27 October 2023).
16. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2023.
17. Oksuz, K.; Cam, B.C.; Kalkan, S.; Akbas, E. Imbalance problems in object detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 3388–3415. [CrossRef] [PubMed]
18. Prusty, S.; Srikanta, P.; Sujit, K.D. SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. *Front. Nanotechnol.* **2022**, *4*, 972421. [CrossRef]
19. Wu, Q.; Ye, Y.; Zhang, H.; Ng, M.K.; Ho, S.-S. ForesTexter: An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Syst.* **2014**, *67*, 105–116. [CrossRef]
20. Arlot, S.; Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **2010**, *4*, 40–79. [CrossRef]
21. Tsoumakas, G.; Katakis, I. Multi-label classification: An overview. *Int. J. Data Warehous. Min. (IJDWM)* **2007**, *3*, 1–13. [CrossRef]
22. Trohidis, K.; Tsoumakas, G.; Kalliris, G.; Vlahavas, I. Multi-label classification of music into emotions. In Proceedings of the ISMIR R, Philadelphia, PA, USA, 14–18 September 2008; Volume *8*, pp. 325–330.
23. Nigam, P. *Applying Deep Learning to ICD-9 Multi-Label Classification from Medical Records*; Technical report; Stanford University: Stanford, CA, USA, 2016.
24. Yang, B.; Sun, J.T.; Wang, T.; Chen, Z. Effective multi-label active learning for text classification. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009.
25. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28* . [CrossRef] [PubMed]
26. Ciaglia, F.; Zuppichini, F.S.; Guerrie, P.; McQuade, M.; Solawetz, J. Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark. *arXiv* **2022**, arXiv:2211.13523.
27. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13; Springer International Publishing: Cham, Switzerland, 2014.

28. Everingham, M.; Eslami, S.M.A.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
29. Singh, D.; Jain, N.; Jain, P.; Kayal, P.; Kumawat, S.; Batra, N. PlantDoc: A dataset for visual plant disease detection. In Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, Hyderabad, India, 5–7 January 2020; pp. 249–253.