Addressing Long-Tailed Distributions in Object Detection via Sampling, Augmentation, Algorithm-Level and Generative Methods

Adrien Morille
s350501@studenti.polito.it

Finn Schinke
s350507@studenti.polito.it

Abstract

In real-world computer vision tasks, datasets are often inherently imbalanced, causing object detection models to underperform on under-represented classes. This study investigates strategies to mitigate class imbalance, including sampling approaches (class-aware and difficulty-aware sampling), algorithm-level methods (loss reweighting), and data augmentation techniques (mosaic, random, copy-and-paste) and a generative method (GAN-based augmentation). Experiments are conducted on a dice object detection dataset, modified to create controlled imbalanced distributions using Zipfian functions.
Results show that handling class imbalance significantly affects detection performance. Sampling and augmentation methods consistently improve mean Average Precision (mAP@0.50), with GAN-based augmentation achieving the highest single approach performance (63.69%), particularly enhancing detection of rare classes. Loss reweighting shows inconsistent gains, while combinations of strategies exhibit saturation effects with surpassing the best single method.

1. Introduction

In many real-world computer vision tasks, datasets are inherently imbalanced, with certain object classes appearing far more frequently than others. Such imbalance can significantly degrade the performance of object recognition models, particularly for under-represented classes. Effectively addressing this issue is crucial for improving detection accuracy and ensuring robust model generalization.

This paper investigates a range of strategies designed to mitigate class imbalance in object detection. Specifically, we consider four categories of approaches: sampling methods, algorithm-level methods, generative approaches and data augmentation techniques. For sampling, we employ class-aware sampling and difficulty-aware sampling; for algorithm-level adjustment, we apply loss reweighting;for augmentation, we use mosaic augmentation, copy-and-paste augmentation; and for generative approaches we use GAN-based augmentation. Each method is applied to enhance the representation and detectability of under-represented object classes.

With this Paper we want to discuss the following research questions (RQ):

RQ1: How do different strategies for handling class imbalance affect the performance of an object detection model in a controlled, real-world-inspired scenario?

RQ2: Which category of imbalance-handling strategies achieves the largest performance improvements on highly imbalanced object detection datasets?

RQ3: Do class-imbalance mitigation strategies also improve object detection performance on balanced datasets, particularly for difficult or hard-to-detect classes?

RQ4: Do combinations of multiple imbalance-handling strategies lead to additive performance gains, or do saturation effects occur?

To evaluate and compare the effectiveness of these methods, we conduct experiments in the context of the game Yahtzee, focusing on the accurate detection of dice. The findings of this study contribute to a better understanding of how different imbalance-handling strategies affect object recognition performance and offer guidance for selecting the most effective methods in a possible future application for a online implementation of Yahtzee.

2. Performance Evaluation

Object detection performance depends not only on the dataset but also on careful choices regarding the model architecture, training strategy, and evaluation method. A well-suited model can improve both accuracy and efficiency, while appropriate training parameters help ensure stable learning and good generalization to unseen data. In addition, selecting meaningful evaluation metrics is important for measuring how well the model performs in real-world scenarios.

2.1. Dataset and Model

In this project, we aim to analyze how imbalanced and balanced datasets influence the performance of an object detection model. To obtain meaningful and reliable results, we consider the degree of imbalance and therefore need a way to distinguish between varying degrees of class balance and imbalance. In the literature, the Imbalance Ratio (IR) is commonly used for this purpose [2].

$$IR = \frac{N_{max}}{N_{min}}$$

The literature does not reach a strict consensus on the definition of a class-imbalanced dataset; however, an Imbalance Ratio (IR) of at least 1.5 is frequently cited as a practical threshold. We employ this metric throughout the following sections to evaluate and compare our datasets. To ensure a rigorous baseline, we selected a standard model that does not incorporate specific optimization methods for long-tailed distributions. Additionally, given our computational constraints, we required an architecture compatible with pretrained weights to facilitate efficient transfer learning.

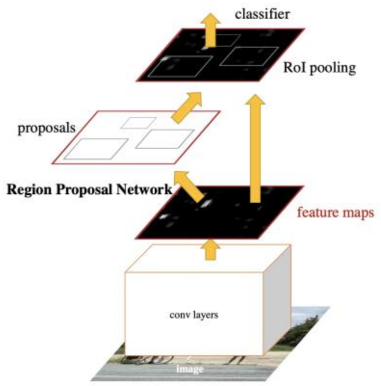

Figure 1: Structure of Faster R-CNN [1]

Consequently, we chose Faster R-CNN with a ResNet50-FPN backbone. This architecture offers a robust trade-off between detection accuracy and inference speed. In this configuration, ResNet50 serves as the primary feature extractor, utilizing deep residual learning to mitigate the vanishing gradient problem and capture complex visual patterns. These features are further enhanced by a Feature Pyramid Network (FPN), which constructs a multi-scale feature hierarchy, allowing the model to detect objects of varying sizes more effectively than a standard single-scale backbone.

The resulting multi-scale feature maps are fed into a Region Proposal Network (RPN), which identifies candidate object regions by estimating objectness scores and bounding box offsets for predefined anchors. These proposals are subsequently refined through RoI Align to extract fixed-size feature representations. Finally, these features are processed by the detection head for class prediction and bounding box regression. This integration preserves the superior localization and detection capabilities of Faster R-CNN while benefiting from the powerful representation learning of ResNet50 and the scale-invariance provided by the FPN.

2.2. Training and Evaluation

Training is performed for 5 epochs with batch size 16, learning rate 0.005, SGD optimizer with momentum 0.9 and weight decay 0.0005, and StepLR scheduler. Images are adaptively resized between 800 and 1333 pixels.

Model evaluation is performed using the mean Average Precision (mAP) metric at IoU threshold 0.5, following the PASCAL VOC protocol. The evaluation process begins by running inference on the balanced test set, which contains 25 images per class to ensure fair per-class assessment. During inference, predicted bounding boxes are filtered using a confidence threshold of 0.05 to retain high-confidence detections while minimizing false positives.

For each class, Average Precision (AP) is calculated by matching predicted boxes to ground truth boxes based on IoU. A prediction is considered a true positive if its IoU with a ground truth box exceeds 0.5 and that ground truth has not been previously matched. Predictions are sorted by confidence score, and precision-recall curves are computed from the cumulative true positives and false positives. AP is then calculated using 11-point interpolation across recall levels from 0 to 1. The final mAP is the arithmetic mean of AP values across all six dice classes, providing a single metric that captures overall detection performance while maintaining sensitivity to per-class accuracy.

All experiments were conducted on Google Colab with NVIDIA Tesla T4 GPU (16GB memory). Training time per configuration ranges from 20 to 40 minutes depending on augmentation complexity, with the complete study requiring approximately 20 GPU hours.

3. Dataset and Dataset Creation

The dataset used in this work is the dice object detection dataset from Roboflow Universe[1], which contains 1,459 annotated images labeled for object detection. It comprises six distinct classes, corresponding to the face values of standard six-sided dice (labels “1” through “6”), so that each detected object represents a single dice face in the

[1] https://universe.roboflow.com/workspace-spezm/dice-0sexk

image. The annotations provide bounding boxes around each visible dice face, enabling supervised training of detection models to locate and classify dice results. This dataset is publicly available under a Creative Commons Attribution (CC BY 4.0) license and can be exported in common input formats for training object detection models such as COCO format or YOLO format. The original dataset is nearly perfectly balanced. In the following you can see the distribution of the number of annotations of each class.

Ann. of Class	*1*	*2*	*3*	*4*	*5*	*6*
Training	442	455	427	434	454	579
Val.	59	59	69	69	59	69
Test	35	28	41	36	25	35

Table 1: Original Class Distribution

The IR of the original dataset is 1.31 for the training, 1.16 the validation and 1.64 the test. Thus, we observe that the original dataset used for training and validation is balanced, whereas the test dataset cannot be considered balanced. In the next segment of this work, we will therefore further adjust this dataset.

3.1. Data Processing

We modified our dataset in two different ways to address class imbalance. In the first approach, we balanced the dataset by removing labels from certain classes, thereby reducing the dominance of over-represented categories. In the second approach, we attempted to remove entire images along with all associated labels to avoid introducing false negatives, which can occur when an image contains one or more minority class instances that are lost due to label deletion.

To systematically study the effects of class imbalance on object detection performance, we plan to induce controlled long-tailed distributions in our datasets using a Zipfian function [4].

$$p(n) = \frac{1}{n}$$

The $p(n)$ shows the probability of the number of annotations of each class in the dataset. A Zipfian distribution follows a power-law pattern in which a few classes occur very frequently while many others occur rarely, which closely resembles the imbalance patterns observed in real-world data and benchmarks designed for imbalance research [4]. Because the Zipfian distribution provides a parametric and interpretable way to control the degree of imbalance (through its exponent parameter), it allows us to generate and compare datasets with varying imbalance severity in a principled manner, rather than arbitrarily subsampling classes. This approach ensures that our experimental results reflect not only the presence of imbalance, but also how different imbalance patterns impact the learning behavior of object detection models. The perfect balanced set of our original dataset will have 427 training, 59 validation and 25 test annotations, because these are maximum possible annotations that each class has. The perfect Zipfian Distribution would look like the following.

Ann. of Class	*1*	*2*	*3*	*4*	*5*	*6*
Training	71	85	107	142	214	427
Val.	10	12	15	20	30	59

Table 2: Annotations of Zipfian Distribution

To ensure comparability of the results, we will consistently use the same balanced test dataset across all experiments.

3.1.1 Annotation Based Processing

To balance the dataset using the annotation-based approach, we first defined a target number of annotations per class. We then removed annotations from the dataset, accordingly, resulting in the following class distribution.

Ann. of Class	*1*	*2*	*3*	*4*	*5*	*6*
Training	427	427	427	427	427	427
Val.	59	59	59	59	59	59
Test	25	25	25	25	25	25

Table 3: Ann. Balanced Class Distribution

With this we achieved a totally balanced dataset with an IR of 1 for each dataset. For the creation of the annotation-based imbalanced dataset, we used the target values defined by the Zipfian function on the newly created balanced dataset and removed annotations accordingly to achieve the desired class distribution.

Ann. of Class	*1*	*2*	*3*	*4*	*5*	*6*
Training	71	85	107	142	214	427
Val.	10	12	15	20	30	59
Test	25	25	25	25	25	25

Table 4: Ann. Imbalanced Class Distribution

This new Dataset follows strictly the Zipfian Distribution, and we are getting a IR of 6.0 for the training and 5.9 for the validation. Therefore, we introduced a high imbalance into the dataset.

3.1.2 Image Based Processing

For the image-based processing, the previously defined target values were first loaded. Subsequently, images were randomly sampled from the original dataset and added until the target values were approximated as closely as possible.

In this way, the newly constructed balanced dataset exhibits the following values:

Ann. of Class	1	2	3	4	5	6
Training	427	427	427	427	427	456
Val.	59	59	59	59	59	59
Test	25	25	25	25	25	25

Table 5: Image Balanced Class Distribution

Due to multiple object in one picture, we don't get a perfect balanced set but with an IR of 1.07 for the training and 1 for the validation we get a much better result as the original dataset. For the creation of the image-based imbalanced dataset, we used the target values defined by the Zipfian function again.

Ann. of Class	1	2	3	4	5	6
Training	109	109	182	182	212	425
Val.	28	28	57	57	28	57
Test	25	25	25	25	25	25

Table 6: Image Imbalanced Class Distribution

With this we are getting an IR of 3,9 for the training and 2,04 for the validation. This means we have again an imbalanced dataset.

3.1.3 Dataset Decision

For the remainder of this paper, we utilize the image-based datasets, even though they are less precisely aligned with the target values. This approach is chosen to reduce the number of false negatives, which could otherwise have a substantial impact on the overall performance of the model. While both datasets are inherently imbalanced according to the IR definition, they remain well-suited for the subsequent analyses.

4. Optimization of Imbalanced Dataset

To assess the most effective strategies for improving model performance on imbalanced datasets, it is essential to identify the specific type of imbalance present. In our study, the imbalance exists across classes, all of which appear in the foreground of the images, giving rise to a foreground–foreground class imbalance [5]. Approaches to address this type of imbalance are generally classified into three categories: sampling approaches, algorithm-level approaches, and data augmentation approaches [6]. To complement these methods, we additionally implemented generative approaches. In the following sections, we provide a detailed discussion of each category and demonstrate the application of representative methods from each to our dataset. Furthermore we compare the performance of each approach that ware shown in the Appendix.

4.1.1 Sampling approaches

In object recognition, sampling approaches aim to balance the training data by adjusting the frequency of object classes. In our study, we employ class-aware sampling, which ensures that under-represented classes are selected more frequently during training, and difficulty-aware sampling, which prioritizes samples that are harder for the model to recognize. These strategies allow the detector to better learn rare or challenging object instances, while accounting for the co-occurrence of multiple objects within a single image [6].

Class-Aware Sampling

Class-aware sampling or Repeat factor sampling (RFS) is a data-level technique that adjusts the probability of selecting samples from each class during training. Classes that are under-represented are sampled more frequently, ensuring the model encounters them more often. This helps improve detection performance for rare object categories without altering the original image content.

Our implementation builds a dictionary mapping each dice class to the list of image indices containing that class. During training, the sampler first uniformly selects a random class, then randomly picks an image from that class's index list. This two-step process ensures equal class representation regardless of the original data distribution. The sampler generates indices for twice the dataset size per epoch, effectively doubling the model's exposure to rare classes while maintaining training efficiency.

In comparison to the imbalanced dataset, an overall performance improvement of 20.48 % is achieved. Notably, the previously underperforming class 5 shows a substantial increase of 63.74 %, highlighting the effectiveness of the applied strategy for this class. The only performance degradation observed occurs in class 4, with a decrease of 6 %. This reduction may be attributed to the effects of class-aware sampling, which modifies the original data distribution and can lead to the over-representation of rare or noisy samples. As a result, the model may be exposed more frequently to hard, ambiguous, or mislabeled instances, thereby increasing the risk of overfitting and negatively impacting generalization performance.

Difficulty aware sampling

Difficulty-aware sampling prioritizes samples that are challenging for the model to classify or detect. By focusing on hard examples, the model learns more discriminative features and improves its ability to detect both under-represented and complex objects, leading to better overall performance on imbalanced datasets.

Our implementation calculates a difficulty score for each dice instance based on four weighted factors: size relative to image area (smaller dice are harder), distance to image edges (edge proximity increases difficulty), density of nearby objects (overlapping dice increase complexity), and estimated occlusion from bounding box intersections. Each image receives an average difficulty score across its instances. The sampler combines class balance probabilities with difficulty scores using a configurable weight parameter (DIFFICULTY_WEIGHT), then samples images with replacement according to the combined probability distribution.

In comparison to the imbalanced dataset, an overall performance improvement of 22.88 % is achieved. Therefore a 2.4 % higher increase than the class aware sampling. For this method the best increase is class 5 with a 66.96 % increased performance. Just for class 2 we can see a decreasing performance of 6.49 % in comparison to the results of the class aware sampling. This is possibly because class-aware sampling directly increases the model's exposure to underrepresented classes, which is particularly beneficial when the primary limitation of a class is data scarcity rather than sample difficulty. In contrast, difficulty-aware sampling prioritizes hard examples across all classes and may insufficiently emphasize a specific class if its samples are not consistently identified as difficult or if perceived difficulty is dominated by noise. Consequently, for class 2, class-aware sampling demonstrates superior performance, as improved representation appears to be more critical than targeted learning from challenging instances.

4.1.2 Algorithm-Level Approaches

Algorithm-level methods address class imbalance by modifying the learning process rather than the dataset itself. We apply loss reweighting, which assigns higher importance to under-represented classes during training, effectively guiding the model to focus more on minority objects. This approach reduces bias towards dominant classes and improves the model's ability to detect less frequent object categories without altering the data distribution [6].

Loss Reweighing

Loss reweighting is an algorithm-level approach that modifies the training loss to give more importance to minority classes. Each class contributes to the loss proportionally to its rarity or difficulty, encouraging the model to focus on under-represented object categories without changing the dataset.

Our implementation replaces the standard cross-entropy classification loss in the Faster R-CNN RoI heads with a custom Focal Loss module. The loss applies the formula

$$FL = -\alpha(1 - pt)^{\gamma} * log(pt)$$

where pt represents the model's predicted probability for the correct class. The alpha parameter balances positive and negative examples, while gamma controls the rate at which easy examples are down-weighted. Box regression continues to use smooth L1 loss unchanged. This modification affects only the classification branch, allowing the model to focus learning capacity on ambiguous or difficult detections.

The Loss Reweighing shows a similar overall performance like the class aware sampling and shows an increase of 22.48 % in comparison of the imbalanced dataset. Even in the class comparison we can see a similar performance to the class aware sampling. The reason could be that both methods address class imbalance by effectively increasing the influence of minority classes during training. Class-aware sampling achieves this by oversampling underrepresented classes, thereby increasing their frequency in the training batches, while loss reweighting assigns higher loss weights to minority-class samples, amplifying their contribution to the optimization process. Although the mechanisms differ, both approaches shift the learning objective toward underrepresented classes, leading to comparable gradients and, consequently, similar model performance when class imbalance is the primary challenge.

4.1.3 Augmentation Approaches

Data augmentation increases the effective representation of minority classes by generating additional training samples. In our work, we utilize mosaic augmentation, copy-and-paste augmentation, and GAN-based augmentation. Mosaic and copy-and-paste augmentations create new images by combining existing object instances, while GANs synthesize entirely new objects. These techniques enhance model robustness and help address class imbalance, although methods like copy-and-paste may require segmentation masks, which can increase annotation effort [6].

Mosaic Augmentation

Mosaic augmentation combines multiple images into a single composite image, placing different object instances together. This increases the diversity of object contexts and scales, exposing the model to a richer variety of appearances and spatial arrangements, which enhances robustness and helps mitigate class imbalance.

Our implementation selects four images (the current index plus three random samples) and arranges them in a 2x2 grid. The split point between quadrants is randomized within ±25% of the center to prevent the model from learning fixed spatial patterns. Each image is resized to fit its assigned quadrant while maintaining aspect ratio, and all bounding box annotations are scaled and translated accordingly. Boxes that become too small after resizing (less than 1 pixel in either dimension) or extend outside the output boundaries are filtered out. The augmentation is applied probabilistically (MOSAIC_PROB parameter) to ensure the model still encounters standard single images during training.

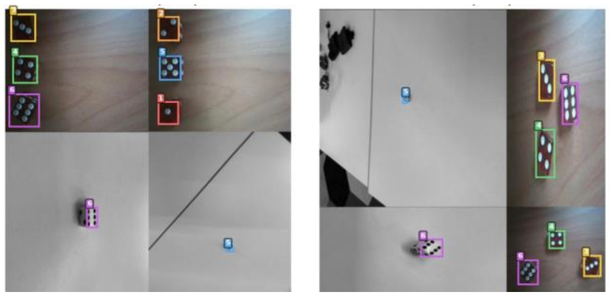
Figure 2: Example Mosaic Augmentation

In comparison to the imbalanced dataset, an overall performance improvement of just 12 % is achieved. Which is less than the performance of the Sampling and the Algorithm-Level Approaches. Mosaic augmentation combines multiple images into a single training sample, which can distort object scale, spatial context, and class-specific features, particularly for small like our dices. This may hinder the model's ability to learn stable and discriminative representations. Furthermore, mosaic augmentation does not explicitly address class imbalance; therefore, it cannot compensate for skewed class distributions in the same way as class-aware sampling, difficulty-aware sampling, or loss reweighting. In contrast, these methods directly modify the sampling or optimization process to emphasize underrepresented or informative samples, resulting in more effective learning and superior performance.

Random Augmentation

Random augmentation refers to a class of data augmentation techniques that apply stochastic transformations to input images, such as flipping, rotation, scaling, cropping, or color jittering. The primary goal is to increase the diversity of the training data, improve model generalization, and reduce overfitting without explicitly creating new object instances.

Our implementation applies two categories of random transformations with 50% probability each. The first is horizontal flipping, which mirrors the image and correspondingly adjusts all bounding box x-coordinates to maintain annotation accuracy. The second is color jittering, which randomly modifies brightness, contrast, and saturation within ±20% of original values. These transformations are applied after any mosaic or copy-paste augmentations, ensuring they affect the final composite image. The transformations preserve spatial relationships and object scales while introducing variability that helps the model generalize to different lighting conditions and orientations.

Random augmentation shows, in comparison to the imbalanced dataset, an overall performance improvement of 14.93%, which is nearly 3% higher than that achieved with mosaic augmentation. This gain is primarily driven by a substantial performance increase of 38.94% in class 5 compared to mosaic augmentation. This improvement can be attributed to the fact that random augmentation preserves the original spatial structure, scale, and contextual relationships of objects while still introducing variability through transformations such as flipping, scaling, or color jittering. Such properties are particularly beneficial for class 5, whose discriminative features rely on fine-grained details.

Copy and Paste Augmentation

Copy-and-paste augmentation inserts object instances from one image into another, effectively creating new training examples for minority classes. This method improves the representation of rare objects in diverse contexts, although it often requires segmentation masks to accurately extract and place objects.

Our implementation randomly selects source images and extracts individual dice instances using their bounding boxes with a small padding. Each extracted instance undergoes random transformations: scaling (80-120%), rotation (±15 degrees), and brightness/contrast adjustment. The augmentation then attempts to place instances on the target image using a collision-detection algorithm that checks IoU with existing boxes to prevent overlap. Gaussian blur is applied to the paste mask edges to create natural blending with the background. The number of instances to paste (1-4) is randomized, and the system makes up to 20 placement attempts per instance before skipping. All annotations are updated to include the newly pasted objects with their transformed bounding boxes.

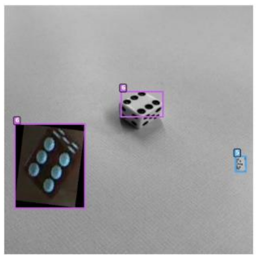
Figure 3: Example Copy and Paste Augmentation

In comparison to the imbalanced dataset, an overall performance improvement of 17.65 % is achieved. Which is again an increase of around 3 % in comparison to the random augmentation. This can be due to its ability to explicitly increase object instance diversity, particularly for rare classes. Unlike random augmentations, which primarily alter image appearance without increasing object counts, and mosaic augmentations, which combine multiple images but may distort object size and context, Copy-Paste preserves realistic object scales and locations while introducing novel instances into varied backgrounds.

4.1.4 Generative Approaches

Generative approaches in computer vision aim to synthesize new data samples that resemble the underlying distribution of a given dataset. These methods, often based on Generative Adversarial Networks (GANs) [7].

GAN

GAN-based augmentation uses generative adversarial networks to synthesize new object instances or entire images. These synthetic samples increase the diversity of under-represented classes and allow the model to learn features that may be scarce in the original dataset, improving detection performance for rare objects.

Our implementation uses Conditional Deep Convolutional GAN (cDCGAN) architecture. The Generator network (3.99M parameters) consists of five transposed convolutional layers with batch normalization and ReLU activations, taking a 100-dimensional noise vector concatenated with a 50-dimensional class embedding and progressively upsampling to produce 64x64 RGB images via a final Tanh activation. The Discriminator network (2.79M parameters) uses five convolutional layers with LeakyReLU activations and batch normalization, receiving the image concatenated with a spatial label embedding and outputting a real/fake probability through a Sigmoid activation. Training is performed for 500 epochs using Binary Cross-Entropy loss with Adam optimizer (learning rate 0.0002, beta1=0.5). For scene generation, we extract background images from the original dataset and place GAN-generated dice at random positions, using collision detection to avoid overlapping with existing objects. Each generated dice is randomly scaled between 60-120 pixels and multiple dice (1-4) are placed per scene. Training and validation use these synthetic images while testing is performed exclusively on real Roboflow images to verify generalization to authentic data.

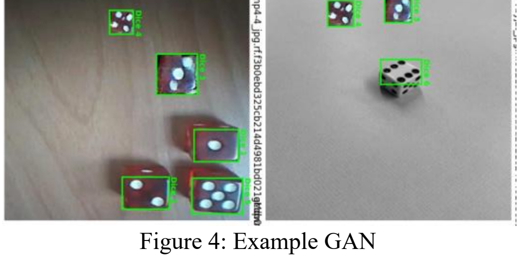

Figure 4: Example GAN

In comparison to the imbalanced dataset, an overall performance improvement of 23.1 % is achieved. Which is the biggest improvement of all the implemented approaches. This is probably due to their ability to generate highly realistic and diverse samples that closely follow the underlying data distribution. Unlike traditional augmentations, which rely on simple transformations or object recombination, GANs can create entirely new instances, including rare or underrepresented classes, while preserving complex structures, textures, and context.

4.1.5 Combination of Methods

We combine different approaches to leverage their complementary strengths: sampling addresses class imbalance by increasing the exposure of rare or difficult classes, while augmentation enhances data diversity and robustness, improving the model's generalization. Specifically, we integrated Class-Aware or Difficulty-Aware Sampling with Mosaic and Random Augmentation because sampling alone improves class representation but does not increase visual variability, and augmentation alone increases diversity but cannot correct for under-represented classes. By combining both, the model encounters rare and challenging instances more frequently and in a wider range of contexts. This integration leads to higher performance than individual methods, with Class-Aware Sampling + Mosaic + Random achieving 65.3% mAP50 and Difficulty-Aware Sampling + Mosaic + Random reaching 66.8% mAP50, demonstrating more stable and generalized detection across all classes.

5. Optimization of Balanced Dataset

In addition to experiments conducted on imbalanced datasets, we also apply the same imbalance-handling strategies to the balanced dataset. This additional evaluation allows us to analyze whether the observed improvements are specific to imbalance mitigation or whether the methods also enhance performance in already balanced scenarios. This comparison enables us to determine whether the applied strategies improve detection of difficult or ambiguous instances independent of class distribution effects.

Due to its strong performance in previous experiments, we applied the Difficulty-Aware Sampling approach as well as a combined strategy consisting of Difficulty-Aware Sampling, Random Augmentation, and Copy-and-Paste Augmentation to the baseline (balanced) dataset. When using Difficulty-Aware Sampling alone, we observed a performance improvement of 3.77%, compared to a substantially larger gain of 22.88% obtained on the imbalanced dataset. This result indicates that the effectiveness of imbalance-handling methods is considerably reduced when class distributions are already balanced, suggesting that such techniques primarily address distributional bias rather than providing equally strong benefits in balanced scenarios.

6. Experimental Results

This section presents the results of our experiments structured around the four research questions defined in Section 1.

6.1. RQ1: Impact of approaches

The results show that different class-imbalance handling strategies substantially affect object detection performance. The baseline model achieves an overall mAP50 of 77.8%, while training on an imbalanced dataset without correction degrades performance to 40.59%. With the introduction of a single approach, we achieved a performance increase of 23.1% (GAN 63.69%) and with a mix of methods we achieved an increase of 26.22% to 66.81%. With these results we can see that the different approaches provide a large performance increase.

6.2. RQ2: Comparison of approach categories

Data-level strategies effectively mitigate class imbalance, with Class-Aware Sampling and Difficulty-Aware Sampling improving mAP50 to 61.1% and 63.5%, respectively. Algorithm-level loss reweighting achieves a comparable gain (63.1%). Classical augmentations, including Mosaic (52.6%), Random Augmentation (55.5%), Copy-and-Paste (58.2%), and GAN-based augmentation (63.7%), further enhance performance, particularly for under-represented or challenging classes.

6.3. RQ3: Performance of class-imbalance mitigation strategies on balanced dataset

On balanced datasets, class-imbalance mitigation strategies lead to modest performance improvements. While the gains are smaller than those observed on imbalanced datasets, the methods still offer slight benefits, particularly for challenging or difficult-to-detect instances.

6.4. RQ4: Performance of mixed approaches

Mixed approaches that combine sampling and augmentation strategies provide additional performance gains over individual methods. Class-Aware or Difficulty-Aware Sampling integrated with Mosaic and Random Augmentation improves mAP50 to 65.3% and 66.8%, respectively, surpassing standalone augmentations and sampling. These results indicate that combining targeted class exposure with diverse data transformations enhances the detector's ability to generalize across both under-represented and challenging object instances.

7. Conclusion

This study demonstrates that class imbalance significantly impacts object detection performance and that different strategies vary in effectiveness. Sampling approaches consistently improve performance over imbalanced baselines, while algorithm-level methods such as loss reweighting do not work well when combined with other methods, often showing inconsistent gains or saturation effects when integrated into a multi-strategy pipeline. Among all strategies, GAN-based augmentation achieves the highest single approach improvement, particularly for rare classes, highlighting its ability to generate realistic and diverse samples that enhance model generalization. Combining multiple methods offers limited additional benefits, indicating that once imbalance is effectively addressed, further strategies provide diminishing returns. These findings provide practical guidance for selecting imbalance-handling techniques in real-world object detection tasks and for a possible game adaptation for Yahtzee.

References

[1] S. Ren, K. He, R. Girshick and J. Sun (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, doi: 10.1109/TPAMI.2016.2577031.

[2] S. Zhao, J. Gui (2025). A Survey on Small Sample Imbalance Problem: Metrics, Feature Analysis, and Solutions. doi: 10.48550/arXiv.2504.14800

[3] Kraiem, M. S., Sánchez-Hernández, F., & Moreno-García, M. N. (2021). Selecting the Suitable Resampling Strategy for Imbalanced Data Classification Regarding Dataset Properties. An Approach Based on Association Models. Applied Sciences, 11(18), 8546. https://doi.org/10.3390/app11188546

[4] Zhu, Yueying & Zhang, Benwei & Wang, Qiuping & Li, Wei & Cai, Xu. (2018). The principle of least effort and Zipf distribution. Journal of Physics: Conference Series. 1113. 012007. 10.1088/1742-6596/1113/1/012007.

[5] K. Oksuz, B. Cam, S. Kalkan, E. Akbas (2020). Imbalance Problems in Object Detection: A Review. https://doi.org/10.48550/arXiv.1909.00169

[6] N. Crasto (2024). Class Imbalance in Object Detection: An Experimental Diagnosis and Study of Mitigation Strategies.

[7] I. Goodfellow (2014). Generative Adversarial Networks. arXiv:1406.2661

8. Appendix

A1. Results of the Approaches

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
map50	77.80%	40.59%	61.07%	63.47%	63.07%	52.59%	55.52%	58.24%	63.69%	65.27%	66.81%
maAP50_1	74.21%	41.07%	48.80%	55.79%	57.91%	46.11%	38.87%	45.85%	61.40%	44.98%	61.26%
mAP50_2	93.17%	54.26%	79.22%	72.73%	72.73%	69.51%	63.99%	70.55%	72.73%	72.73%	72.73%
mAP50_3	58.79%	32.08%	39.15%	39.54%	37.93%	44.50%	31.43%	37.71%	36.21%	50.17%	49.93%
mAP50_4	83.45%	51.58%	45.58%	52.71%	52.79%	44.19%	52.59%	55.76%	52.09%	63.08%	52.91%
mAP50_5	89.29%	24.69%	88.43%	91.65%	91.96%	47.68%	86.62%	75.44%	94.44%	86.85%	88.24%
mAP50_6	67.90%	39.86%	65.26%	68.43%	65.08%	63.56%	59.61%	64.12%	65.27%	73.81%	75.80%

	12.	13.
map50	81.57%	81.42%
maAP50_1	80.07%	83.07%
mAP50_2	94.41%	96.92%
mAP50_3	65.27%	65.31%
mAP50_4	72.87%	70.02%
mAP50_5	94.93%	99.33%
mAP50_6	81.85%	73.88%

1. Baseline
2. Imbalanced Dataset
3. Class-Aware Sampling
4. Difficulty-Aware Sampling
5. Loss Reweighting
6. Mosaic Augmentation
7. Random Augmentation
8. Copy-and-Paste Augmentation
9. GAN-Based Augmentation
10. Class-Aware Sampling + Mosaic Augmentation + Random Augmentation
11. Difficulty-Aware Sampling + Mosaic Augmentation + Random Augmentation
12. Difficulty-Aware Sampling on Baseline
13. Difficulty-Aware Sampling + Random Augementation + Copy-and-Paste Augmentation und Baseline