Blog Post Group 18:
The Impact of Realistic Data Augmentation on Waste Detection

Bryan He: 4815300, b.y.he@student.tudelft.nl,
Irtaza Hashmi: 4829360, i.hashmi@student.tudelft.nl,
Mike Wu: 6105041, T.Y.M.Wu@student.tudelft.nl

July 1, 2024

In this blog post we present the results of our own chosen project for the course CS4245 Seminar Computer Vision by Deep Learning 23/24 at TU Delft. The code used written for pre-processing, augmenting, training and evaluating the object detectors can be found in this GitHub repository. The dataset used for this project is called WaRP-D[5] and is available on Kaggle.

1 Introduction

Disposal of garbage in landfills contributes to the global threat of pollution and microplastics buildup in the environment[2]. In theory, recycling waste contributes to reducing environmental pollution, but in reality, not all recyclable waste is recycled due to many challenges present in the pipeline of tasks that comprise recycling. One important subtask in this pipeline is sorting the different types of waste. Sorting waste is difficult for humans but it is necessary as there are countless types of plastics, each made with different materials, making it chemically impossible to recycle when not separated. This is where object detectors from computer vision can be of help. They can be trained to detect the different types of recyclable waste on the conveyor belt, so that robotic arms can do the actual sorting. Object detectors are typically trained on datasets labeled by humans. However, because waste detection is difficult for humans, the resulting dataset consists of many simple instances that are easy for humans to label. This results in the object detector being trained on data that does not fully reflect real-world scenarios. In real scenarios, recyclable waste is never perfectly laid on the conveyor belt; it is usually partially occluded by other waste which makes accurate detection more challenging.

In this project, we conduct experiments with various data augmentation strategies that introduce partial occlusions to the waste that needs to be detected. The goal of this project is to determine whether simulating real-world conditions through these augmentations enhances the robustness of object detectors against occlusions, which are more common in real-world scenarios. If it does, then it contributes to more effective recycling and reduction of environmental pollution.

2 Dataset

This project uses the WaRP (Waste Recycling Plant) [5] dataset, which contains 10.000 HD images that can have 28 different types of recyclable objects in them. WaRP consists of 3 separate parts: WaRP-D (object detection), WaRP-C (classification) and WaRP-S (segmentation). We will only use the WaRP-D dataset to train our object detector.

A ground truth label for an image is a text file with the same name as the image file, but with a `.txt` extension. Each line in the label file represents a bounding box for one of the objects in the image. The format of a line is as follows: `class x_center y_center width height`. `class` is an integer from 0 to 27 that represents one of the types of recyclable objects. `x_center` and `y_center` are the normalized coordinates of the center of the bounding box, which are in the range [0, 1] with (0, 0) being the top-left corner of the image and (1, 1) the bottom-right corner of the image. `width`

Figure 1: Example of an image in the dataset with its objects and corresponding bounding boxes in red. The class that belongs to the bounding box is also displayed above each box.

and `height` are the normalized width and height of the bounding box. An example image from the dataset can be seen in Fig. 1.

The WaRP-D dataset consists of 2452 training samples and 522 test samples. We chose to split the 2452 training samples into another 2 datasets: 80% for training (1962 samples) and 20% for validation (491 samples). The 522 test samples is left as is and will be used as the test set. The reason why this extra split is necessary is so we can have a validation set that can be used during training for tuning hyperparameters such as learning rate, weight decay etc.

2.1 Preprocessing

The images in the WaRP-D dataset come in two different resolutions. Some of the images are 1920x1080 and others are 960x540. To train the model properly, we need to make sure that all the samples are of the same resolution. We have several options:

- Only training using the high resolution samples (1920x1080)
- Only training using the low resolution samples (960x540)
- Down-scaling the high resolution samples to the lower resolution (960x540) and training on the combined set
- Up-scaling the lower resolution samples to the higher resolution (1920x1080) and training on the combined set

To objectively choose one of those options, we compute the objective image quality using the Perception based Image Quality Evaluator (PIQE) metric which measures distortion. A lower PIQE indicates a higher image quality and is thus better.

Set	Average PIQE score
High resolution only	18.94
Low resolution only	24.76
Down-scaling to lower resolution	8.24
Up-scaling to higher resolution	47.56

Table 1: The average PIQE scores of the different sets containing different resolutions.

From Table 1 we can see that down-scaling the higher resolution images (1920x1080) to (960x540) has the best average PIQE score, so to create the dataset we just down-scale all of the high resolution images. All samples are now of resolution 960x540. Doing this also has the added benefit of less GPU memory usage per image during training, which allows us to increase the batch size.

2.2 Data Augmentations

We apply different kinds of augmentations to the training data to simulate real-world conditions more accurately. These augmentations introduce partial occlusions to the objects that are to be detected. Our chosen object detector (see section 3) are fine-tuned on the various datasets created with the augmentation strategies described below. These datasets were pre-created before fine-tuning which has potential drawbacks that we reflect on in section 4.

2.2.1 Cutout

Cutout [1] is a regularization method where a random part of the image is occluded by a square of a fixed size. Training on a dataset on which cutout has been applied may improve the overall performance and robustness of CNNs. Performing cutout on the WaRP-D may make the object detection scenarios more realistic, because it resembles other objects "covering" the object to be detected.

In the object detection case, instead of covering a random part of the image, we cover a random part of the objects of interest. According to the original cutout paper [1], the area of the cutout rectangle is more important than the shape of the rectangle. It also mentions that letting the rectangle fall outside of the ground-truth bounding box is beneficial to performance. For this project, we decided to let the black rectangles have an area that is 30% of the area of the bounding box of the object it should occlude. Given an image x_A from the training set and its ground truth label y_A, a cutout version x_B with label y_B is created as follows: for each bounding box $bb_i = (x_i, y_i, w_i, h_i)$ in ground truth label y_A, randomly place the center of a black rectangle that has an area of $0.30 \cdot w_i \cdot h_i$ (where w_i and h_i are the width and height of bounding box bb_i) inside the patch of image x_A enclosed by bb_i, allowing black rectangle to cross the boundaries of bb_i. The ground truth label of cutout image y_B is equal to y_A. An example can be seen in Fig. 2.

The resulting "CutOut" training dataset consists of the 1962 unmodified samples from the original dataset, plus an additional 491 images which are sampled from the training set (25% of 1962) on which cutout has been applied to.

Figure 2: Example of cutout.

2.2.2 CutMix

CutMix [6] is a regularization method where we replace the occluded region of the image with a patch from another image. This ensures that no information is lost by replacing the image with black pixels

or random noise, like in cutout. Additionally, in the original paper, the ground truth labels are also mixed in proportion to the area of patches in the images. This technique generates more natural images which when trained on, increases the localization ability of the model. Training on a dataset on which CutMix has been applied can improve the performance and robustness of CNNs. Performing CutMix on the WaRP-D dataset can capture more realistic scenarios, where the other objects "are on top" of the object of interest to be detected.

Similarly to cutout, we cover a random part of the object with a patch from another object. We randomly select two training samples and their ground truth label from the training set (x_A, y_A) and (x_B, y_B) to create a cutmix image x_C and its label y_C. We remove a rectangular area of 30% from a bounding box in training sample (x_A, y_A), relocate the center of the rectangle to a random location within the original bounding box, crop a patch from the second training sample (x_B, y_B) with an area equal to the removed patch, and paste it to the x_A. This is done for every labeled object within the image x_A. In other words, image x_C is x_A with some patches sampled from image x_B placed near/on the objects of interest in x_A. Since CutMix is originally designed for classification datasets and not for object detection datasets like WaRP-D, so there are many options for deciding how to contruct the label y_C. For this project we let the label y_C be the same as the label y_A. An example of a CutMix augmented image can bee seen in Figure 3

The "CutMix" dataset we created consists of the 1962 unmodified samples from the original dataset, plus 491 cutmixed samples.

Figure 3: Example of CutMix.

2.2.3 MixUp

MixUp [7] was originally designed to help the problem of memorization and sensitivity to adversarial examples in deep neural networks. It is also a regularization technique that can be applied to our use-case. This technique combines two images together by mixing their pixel values with a certain weight. This can make object detection scenarios more realistic. For example, some materials such as translucent plastic are see-through meaning you might be able to see other object behind it. We can use mixup to simulate this scenario.

To create a mixup image x_C and its ground truth label y_C, we sample two images from the training set and their ground truth label: (x_A, y_A) and (x_B, y_B). The hyperparameter λ is the weight with which the pixel values from x_A and x_B are multiplied before adding them up to obtain image x_C. In other words, pixel values p_A of image x_A are weighted by λ and the values p_B of x_B are weighted by $(1-\lambda)$. To obtain the pixel values p_C of the mixup image x_C augmented image we simply sum the two terms: $\lambda \cdot p_A + (1-\lambda) \cdot p_B$. The ground truth label y_C for the mixup image is simply the concatenation of all the labels y_A and y_B. An example can be seen in Fig. 4.

This "MixUp" dataset has 1962 unmodified samples from the original dataset, plus 491 samples with mixup applied.

Figure 4: Example of MixUp.

Figure 5: Example of Bernoulli.

2.2.4 Uniform

The previous three augmentation strategies were employed to construct an additional dataset we call "Uniform". The 'Uniform" dataset is composed of 1962 samples from the original unmodified training set, and 491 augmented samples. Of these augmented samples, 33.33% consist of mixup, 33.33% of cutout and 33.33% of mixup images. Every augmented sample is augmented by exactly one of the three strategies.

2.2.5 Bernoulli

The "Bernoulli" dataset we created also involves a certain amount of randomness. It consists again of 1962 unmodified samples from the original dataset, and 491 extra samples. These extra 491 samples can have have no augmentation at all, or at most 3 augmentations simultaneously. Augmentations are applied in the following order: first cutmix, then cutout, and lastly mixup (each with 0.5 probability). Fig. 5 shows an example where cutout, cutmix and mixup are applied.

3 Fine-Tuning and Evaluating the YOLOv8 Object Detector

3.1 The YOLOv8 Architecture

The object detector used in this project is YOLOv8[4]. There are multiple variants of YOLOv8 varying in the number of parameters. The variants with a bigger number of parameters can improve accuracy in mAP but at the cost of speed. For this project, YOLOv8n was chosen because it has fewer parameters which makes it compatible with limited computational and data resources. Its architecture comprises three main components: the backbone, neck, and head, as shown in Figure 6. The backbone of YOLOv8 uses Conv blocks and C2f blocks to extract feature maps at various spatial resolutions. A Conv block consists of a 2D convolution layer, batch normalization, and a non linear activation function. A C2f block includes a Conv layer to reduce channels, a split operation that divides the features into two parts, several bottleneck layers which are sequential 2D convolutions with optional residual connections, and a concatenation operation that stack the features back together. The neck component produces three different feature maps by combining the different feature maps provided by the backbone through upsampling, concatenation, Conv, C2f, and an SPPF block, which concatenates the output of multiple max pooling operations at different scales to increase the receptive field and improve object detection at various scales. Finally, in the head component, each feature map produced by the neck is provided to its respective detection head. Each detection head has two branches of sequential Conv blocks: one branch outputs a bounding box, and the other classifies the object within the bounding box.

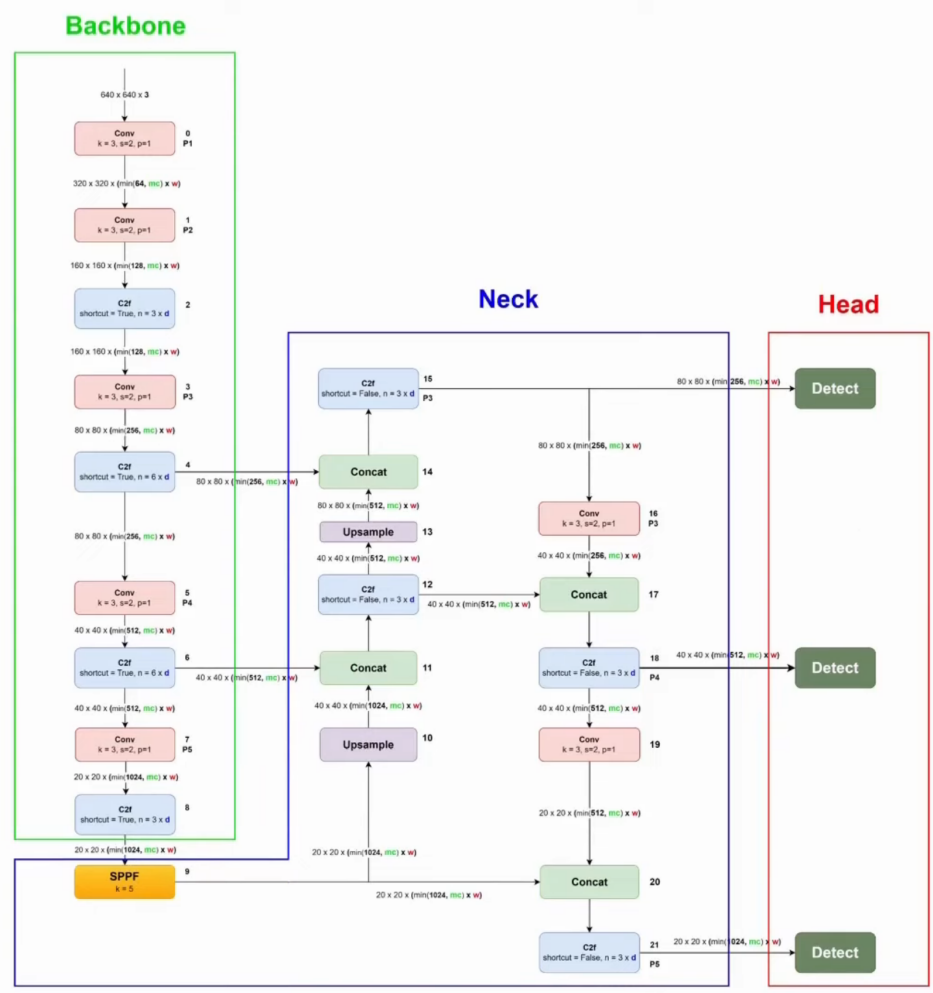

Figure 6: The YOLOv8 architecture *Source: video by Dr. Priyanto Hidayatullah*

3.2 Fine-tuning YOLOv8

For this project, we fine-tuned six different YOLOv8 models, each starting from the same base: a YOLOv8 model pretrained on the COCO dataset[3]. Each of these six YOLOv8 models are fine-tuned on six different versions of the Warp-D dataset described in section 2: original/no augmentation, CutMix, CutOut, MixUp, Uniform, and Bernoulli.

All six models were fine-tuned for 50 epochs. We believe this is sufficient as the the loss curves on the training set and the validation set converge and stabilize around the 50-epoch mark (see Figure 7). Figure 7 consists of three plots, each corresponding to one of the three loss components of YOLOv8's loss function: Binary Cross Entropy loss for classification, and both Distribution Focal Loss and complete Intersection over Union (CIoU) loss for bounding box prediction.

At the 50-epoch mark in Figure 7 it can be seen that the gap between the validation loss and the training loss of YOLOv8 fine-tuned on the original unaugmented dataset is much larger than those of the 5 YOLOv8 models fine-tuned on augmented datasets, with the exception of CutOut. This indicates that data augmentation reduces, but does not completely prevent, overfitting on the training set (again with the exception of CutOut). One possible reason for why YOLOv8 fine-tuned on CutOut overfits more compared to fine-tuning on the original dataset is that the black rectangles on the augmented images are more conspicuous/obvious, so it is a bias that causes YOLOv8 to learn to make more accurate predictions i.e. predicted bounding boxes should be near the black rectangle.

From the loss curves it can be observed that all data augmentation strategies, except for CutOut, result in higher loss values on the training set compared to the original unaugmented dataset. This indicates that the augmentations generally increased the complexity of the detection task, with the exception of CutOut which appears to simplify the task compared to the original dataset.

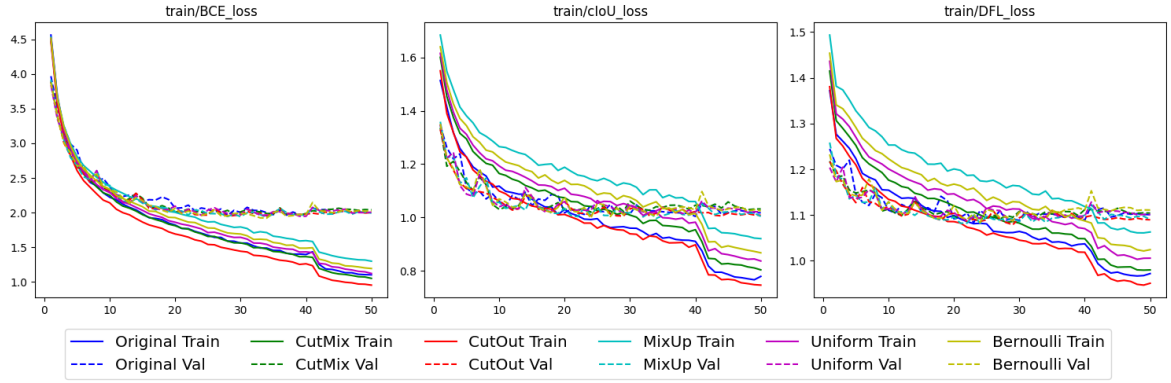

Figure 7: Loss curves on the training set and the validation set for the six fine-tuned YOLOv8 models with different data augmentation strategies. The three plots show the loss components: Binary Cross Entropy (BCE) loss for classification, CIoU loss and Distribution Focal Loss (DFL) for bounding box prediction.

3.3 Performance Evaluation

To assess whether the five data augmentation techniques enhance the accuracy of YOLOv8 on the WaRP-D dataset, we utilize mean Average Precision (mAP), specifically mAP50 and mAP50-95. mAP50 measures the mean average precision of the bounding boxes predicted by YOLOv8 at a single IoU threshold of 0.5, whereas mAP50-95 measures the mean average precision across multiple IoU thresholds ranging from 0.5 to 0.95, so it captures the object detector's ability to accurately localize objects (higher IoU thresholds) and its robustness to less precise predictions (lower IoU thresholds).

Tables 2 and 3 present the mAP of the six fine-tuned YOLOv8 models on the unaugmented test set and a uniformly augmented test set, at IoU thresholds of 50 and 50-95, respectively. In both tables, the mAP score on the unaugmented test set reflects the models' performance in easier and less realistic scenarios, whereas the mAP score on the uniformly augmented test set represents the models' performance in more challenging and realistic scenarios (for description of what is meant by uniform augmentation see section 2.2.4). To evaluate the robustness of the six different fine-tuned YOLOv8 models, we can rank them based on the percentage drop in mAP50 scores when transitioning from

the unaugmented test set to the augmented test set. The ranking from least to most percentage drop, and hence also from most robust to least robust, is as follows: CutMix, Bernoulli, Uniform, CutOut, MixUp, and at last place, the Original/no-augmentation model. All five data augmentation strategies considered in this project outperform the YOLOv8 fine-tuned on the original unaugmented dataset in terms of robustness.

Although YOLOv8 fine-tuned on the unaugmented dataset (Original) outperforms the other data augmented variants in the easier scenario, it fails to generalize to realistic scenarios, as evidenced by significantly larger drops of 47.4% and 53.4% in mAP50 and mAP50-95, respectively, when evaluated on the augmented test set. Interestingly, despite being the most robust (least percentage drop), CutMix does not achieve the highest mAP50 score and is therefore not the most accurate on the augmented test set. In contrast, YOLOv8 combined with the Bernoulli augmentation strategy achieves the highest mAP50 score, making it the most accurate in more challenging and realistic scenarios. The trends observed in Table 2 (mAP50) are consistent with those in Table 3 (mAP50-95). All in all, the observations made here support the statement that simulating real-world conditions through data augmentations can enhance the robustness of object detectors against occlusions which are common in real-world scenario of waste detection.

	test_original	**test_aug**	**%-change**
Original	0.46	0.242	-47.4%
CutMix	0.432	0.289	-33.1%
CutOut	0.447	0.273	-38.9%
MixUp	0.453	0.275	-39.3%
Uniform	0.44	0.284	-35.5%
Bernoulli	0.459	0.304	-33.8%

Table 2: **mAP50 scores** for different data augmentation strategies on the original and augmented test sets. The %-change column indicates the percentage drop in mAP score when transitioning from the original to the augmented test set. In our experiments, CutMix augmentation results in the most robust model i.e. least percentage drop (yellow), and Uniform augmentation is the most accurate on the augmented test set i.e. highest mAP50 score on test_aug (green).

	test_original	**test_aug**	**%-change**
Original	0.35	0.163	-53.4%
CutMix	0.328	0.201	-38.7%
CutOut	0.342	0.19	-44.4%
MixUp	0.345	0.186	-46.1%
Uniform	0.334	0.194	-41.9%
Bernoulli	0.348	0.203	-41.7%

Table 3: **mAP50-95 scores** for different data augmentation strategies on the original and augmented test sets. The %-change column indicates the percentage drop in mAP score when transitioning from the original to the augmented test set. In our experiments, CutMix augmentation results in the most robust model i.e. least percentage drop (yellow), and Uniform augmentation is the most accurate on the augmented test set i.e. highest mAP50-95 score on test_aug (green).

4 Discussion and Limitations

The results of this study support our assumption that data augmentation techniques enhance the robustness of YOLOv8 models when applied to the WaRP-D dataset. The use of mAP50 and mAP50-95 metrics provides a comprehensive evaluation of the models' performance, capturing both the accuracy and robustness of object detection under varying conditions.

From the evaluation, it is evident that data augmentation improves the YOLOv8 models' ability to handle realistic and challenging scenarios. The CutMix and Bernoulli augmentations, in particular, show notable improvements. CutMix, while not achieving the highest mAP50 score, demonstrated

the greatest robustness with the least percentage drop in performance when transitioning from unaugmented to augmented test sets. This indicates that CutMix effectively enhances the model's ability to generalize to diverse and occluded objects, which are common in real-world waste detection scenarios. On the other hand, the Bernoulli augmentation strategy achieved the highest mAP50 score on the augmented test set, indicating that it provides a balanced improvement in both robustness and accuracy. This makes Bernoulli augmentation particularly useful for applications where maintaining high detection accuracy under challenging conditions is critical.

The trends observed in mAP50 are consistent with those in mAP50-95, reinforcing the reliability of these findings. The data suggest that augmentations like CutMix and Bernoulli not only enhance robustness but also help maintain high precision across varying IoU thresholds, which is crucial for practical applications of waste detection.

To clear up ambiguity, it is important to note that our data augmentation technique was done offline on a predefined dataset which can have the undesired consequence of overfitting. Proper data augmentation is done online during training, it should be done dynamically and applied stochastically to the samples in mini-batches sampled by a train loader. We didn't implement proper data augmentation, because we use YOLOv8, which is a third party Python module. Due to time constraints we couldn't extend it with the augmentations considered for this project.

Lastly, despite the promising results, this study has several additional limitations that could undermine our conclusions, which can be addressed in future work:

1. **Datasize and diversity**: The WaRP-D dataset, while useful, may not encompass the full range of variability found in real-world waste scenarios. Larger and more diverse datasets could provide a more comprehensive evaluation of the models' robustness and generalizability. Additionally, the creators of the WaRP-D dataset only has 28 different classes/labels of objects of interest. We also observed that many objects in an image do not have a bounding box in the ground truth label, thus, the dataset we used is actually far from real-world scenarios. The data augmentation techniques can be applied to a more realistic dataset for better results.

2. **Hyperparameters**: We did not experiment with other hyperparameters to fine-tune the model. For example, no grid search was applied. This was due to the time constraints and limited computation resources. It took us days to fine-tune one model and required computational resources that we didn't have. Additionally, we cutout 30% patches from each bounding box, which was fixed. This threshold can be varied and experiments can be run to find the optimal percentage. Lastly, for CutOut and CutMix, we only experimented with occluding each object with exactly one black rectangle or patch, respectively. Future work could experiment whether occluding each object with more than one black rectangle or patch results in more accuracy improvements.

3. **Evaluation Metrics**: While mAP50 and mAP50-95 are robust metrics, incorporating additional evaluation metrics such as F1-score, precision, and recall might provide a more nuanced understanding of model performance, particularly in cases of class imbalance.

5 Contributions

- Bryan He - Finding WaRP-D dataset, writing the high level story line, helping with writing the blog, implement MixUp, writing code for training and evaluating models, and training models.
- Irtaza Hashmi - Implementing the cutmix data augmentation technique, training the model using cutmix dataset, helping with blog skeleton, writing the blog
- Mike Wu - Training model, implementing cutout, writing the blog, helping with blog skeleton

References

[1] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[2] G Lamichhane, A Acharya, R Marahatha, B Modi, R Paudel, A Adhikari, BK Raut, S Aryal, and N Parajuli. Microplastics in environment: global concern, challenges, and controlling measures. *International Journal of Environmental Science and Technology*, 20(4):4673–4694, 2023. Epub 2022 May 26.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[4] Ultralytics. Ultralytics yolov8 github repository. `https://github.com/ultralytics/ultralytics`, 2023. Accessed: 2023-05-22.

[5] Dmitry Yudin, Nikita Zakharenko, Artem Smetanin, Roman Filonov, Margarita Kichik, Vladislav Kuznetsov, Dmitry Larichev, Evgeny Gudov, Semen Budennyy, and Aleksandr Panov. Hierarchical waste detection with weakly supervised segmentation in images from recycling plants. *Engineering Applications of Artificial Intelligence*, 128:107542, 2024.

[6] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[7] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.