DANA4800 – PROJECT TEAMWORK

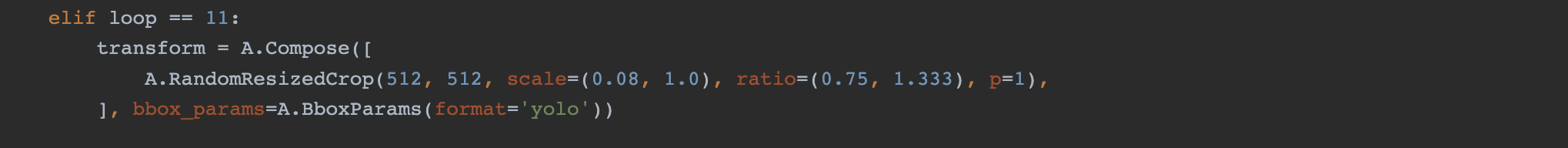
This project works on the prediction of multiple and single categories based on augmentation and running it on different platforms to check and predict using YOLOV8 and PyCharm. Our focus was on categories as mentioned below.

|  |
| --- |
| **Category** |
| Metal lid |
| Non-medication |
| Plastic cup |
| Plastic produce container |
| Plastic single serve |
| Plastic straw |
| Plastic sunscreen bottle |
| Plastic pots and trays |

Firstly, we started this project by collecting pictures from various places and asked help from family and friends for image collection. These images were all primarily collected wherein we clicked pictures from garbage bins, people carrying these items in hand, and it was all amusing. Then we created labels for these images with makesense.ai and submitted same for the assignment 3 last time. Now we ran the same images using augmentation for prediction and have accumulated the results in the drive along with the graphs.

Secondly, for prediction, we first started running the Yolov8 model without running augmentation for all categories mentioned above. These models were trained by storing images on google drive and used google colab to train and predict the results. We had created train, test & val folders already in drive. These were used to train and use as a base for task2. We believe it was because of less sample size and fewer epochs hence we jumped to do it with augmentation and use that model for prediction in step 2.

For step 2, we augmented the train images using PyCharm and stored them back to the train and ran the yolov8 model again. This time the training took a long time to run epochs and sometimes we faced warnings and failures to run the model and had to re-run the model. Additionally, the images took a lot of time to be uploaded on the drive for running the code on google colab and then used for training and prediction. After running the model for augmented images, it started showing prediction with more precision for almost all images and the graphs were also aligned with the results.csv file where almost all had close to 1 showing the great result which we got from running the augmentation. For augmentation along with the code shared we used another augmentation method. The screenshot is pasted below.



For task 2, we started with selecting 4 categories (plastic single serve, plant pots, metal lid & plastic cup) as class 0, class 1, class 2 & class 3 respectively. These are used for task 2 to run, train and test the images to predict. After running the model for epoch 15 on Google colab the system failed, and we had to start the GPU again after only a few hours. Even after doing the same thing, we were not getting the result, so we reduced the epoch to 10 but then again, the GPU failed, and it was an issue after 3 or 4 epochs, so we switched to the traditional method of doing this on Anaconda. On Anaconda there was no issue, but it was taking a lot of time for 10 epochs we reduced the epochs to 5, and then after 6-7 hours finally we got the result. The testing results predicted almost all testing images with more than 60% precision and a few of them were close to 100% even with multiple same categories in the picture. Even the precision table and graphs were updated to increase values and positive graphs. All these codes are stored on google drive along with Python files and CSV files for all with and without augmentation for both task1 & task2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **epoch** | **train/box\_loss** | **train/cls\_loss** | **train/dfl\_loss** | **metrics/precision(B)** | **metrics/recall(B)** | **metrics/mAP50(B)** | **metrics/mAP50-95(B)** | **val/box\_loss** | **val/cls\_loss** | **val/dfl\_loss** | **lr/pg0** | **lr/pg1** | **lr/pg2** |
| **0** | 1.3688 | 2.6391 | 1.4906 | 0.56491 | 0.53007 | 0.53918 | 0.33367 | 1.3384 | 2.3965 | 1.517 | 0.00041534 | 0.00041534 | 0.00041534 |
| **1** | 1.3003 | 1.9591 | 1.4247 | 0.60986 | 0.54382 | 0.53462 | 0.34035 | 1.3965 | 2.2383 | 1.5432 | 0.00066727 | 0.00066727 | 0.00066727 |
| **2** | 1.2466 | 1.6263 | 1.3695 | 0.71479 | 0.54551 | 0.64131 | 0.41552 | 1.3924 | 1.7467 | 1.5159 | 0.0007542 | 0.0007542 | 0.0007542 |
| **3** | 1.1585 | 1.3667 | 1.3122 | 0.72056 | 0.62964 | 0.72003 | 0.47162 | 1.2428 | 1.4438 | 1.4166 | 0.0005075 | 0.0005075 | 0.0005075 |
| **4** | 1.0838 | 1.1785 | 1.2549 | 0.78121 | 0.6758 | 0.77128 | 0.53029 | 1.3079 | 1.1742 | 1.4667 | 0.0005075 | 0.0005075 | 0.0005075 |

Results of Metal Lid as an example and the rest are same.

|  |  |  |
| --- | --- | --- |
| Categories | Without Augmentation | With Augmentation |
| Metal Lid | The model's training losses, including bounding box loss, classification loss, and distance focal loss, decreased progressively, indicating improvement in object localization and classification. The evaluation metrics, such as precision, recall, and mean average precision (mAP), also showed positive trends, demonstrating the model's enhanced ability to detect objects accurately. Overall, the model demonstrated steady improvement during training. | The object detection model underwent 15 training epochs, with decreasing losses over time (e.g., train/box\_loss reduced from 1.5679 to 0.5903). Metrics like mAP50-95(B) improved from 0.49252 to 0.64865, indicating better precision and recall. Learning rates were adjusted gradually (e.g., lr/pg0 started at 0.00065863 and ended at 0.000284). |

Task 2:

Before augmentation had almost no result while with augmentation, we got better result with more than 50%.