EECS 4404/5327 Project Part 5 Report

Rafael Dolores, Luca Fillipelli, Hashir Jamil, Dennis Nevelev

December 4, 2023

Abstract

In response to today's global waste crisis and traditional sorting inefficiencies, two machine learning models leveraging Python's libraries and transfer learning techniques were developed. Beginning with the VGG16 pre-trained network and later advancing to the more precise DenseNet201, this model addresses the critical issue of waste mismanagement and recycling contamination. Complementing this, a multi-label model utilizing Ultralytics' pre-trained YOLOv8n model was trained on a custom, webscraped dataset to classify images with multiple waste objects. Both models were deployed in a web application, enabling real-time, automated waste categorization. The multi-class model's training has yielded a testing accuracy of 71.54% for VGG16 and a markedly higher 86.16% for DenseNet201, with DenseNet also showing improvements across precision, recall, and F1-score metrics. The multi-label model achieved perfect scores (1.00) for all classes at a confidence threshold of 0.960. These results demonstrate the models' effectiveness in accurately classifying single-class and mixed waste items, suggesting that they are effective automated classification systems for waste management.

1 Introduction

1.1 What is your Application?

This application provides a solution to the waste management challenges by offering a user-friendly platform for real-time waste classification through a Streamlit web application. This interface allows users to upload their waste images and have them classified by one of the two distinct neural network models available. The first model, designed for multi-class classification, harnesses the power of DenseNet201 and VGG16 neural networks through pre-training. The second one, for multi-label classification, employs an object detection approach using Ultralytics' YOLOv8n model, which achieves exceptional precision. The application enables users to instantly receive classifications upon upload, thus facilitating efficient recycling practices and contributing to environmental preservation efforts. The source is available publicly at the following GitHub Repository, all information on how to build and run the project are available there.

1.2 What are the Assumptions/Scope of your Project?

The project is inherently limited by the form of information it accepts and the waste classification labels it outputs. The models have been trained on a diverse dataset of images designed to mimic the quality of amateur photos. Therefore, the application implicitly assumes the input data has a corresponding photographic quality. Furthermore, the output labels include various recyclables and non-recyclable materials. The concept of compost, in comparison to general waste, would be indistinguishable from the perspective of the models. Therefore, the scope of the application is limited to the identification of cardboard, glass, metal, paper, plastic and trash.

1.3 Justify Why is your Application Important?

Individuals normally sort through their waste manually. This manual sorting may create inaccuracies in the output thus leading to the mismanagement of waste. This mismanagement leads to increased material waste as otherwise recyclable materials cannot be processed. The classifier will aid in the process of efficient waste management by enabling individuals to automatically classify their garbage without the necessary prerequisite knowledge required for manual classification.

1.4 Similar Applications

TrashBot is a commercial application which uses machine learning to distinguish between various forms of recyclable and compostable waste [1]. Unlike TrashBot, this application focuses on an individual user's experience. Our model better serves the needs of an individual user through the use of a community-centred platform as opposed to the deployment of proprietary technology. Furthermore, the application's open-source nature enables community members to provide contributions to improve upon the base model. The community-centric nature of the model is impossible to achieve with the business-first positioning of TrashBot. Moreover, a significant distinction from TrashBot is this model's capability to handle mixed waste scenarios. It employs multi-label classification, allowing for the identification of various waste categories within a single item [1].

1.5 Adjustments to Part 1 Proposal

The initial project proposal included the possibility of expanding the number of classes to encompass hazardous waste and compost if time permitted. As a high-quality dataset could not be procured for these extra classes, within the provided timeline, they were not included in the final project. Nevertheless, the project was enhanced by introducing a multi-label classification model to address the complexity of real-world waste sorting, where items are often mixed and do not fall neatly into single categories.

2 Methodology

2.1 Design/Pipeline

The architectural diagram in Figure 1 illustrates the machine learning pipeline for image classification with a Kaggle trash dataset and web-scraped Google images. These images undergo preprocessing, including resizing to 224px x 224px pixels for Kaggle-sourced and 640px x 640px for scraped images. This preprocessing involved manual labelling using the LabelImg[2] tool for precise annotation, followed by normalization. The training phase leverages DenseNet201[3] and VGG16[4] for multi-class classification, with the intent of assessing which pre-trained model yields better performance, and the Ultralytics' YOLOv8n[5] for multi-label classification. Both models were refined through Stochastic Gradient Descent and evaluation metrics were used to assess model performance before deployment in the Streamlit web app, which facilitates both single and multiple item waste predictions.

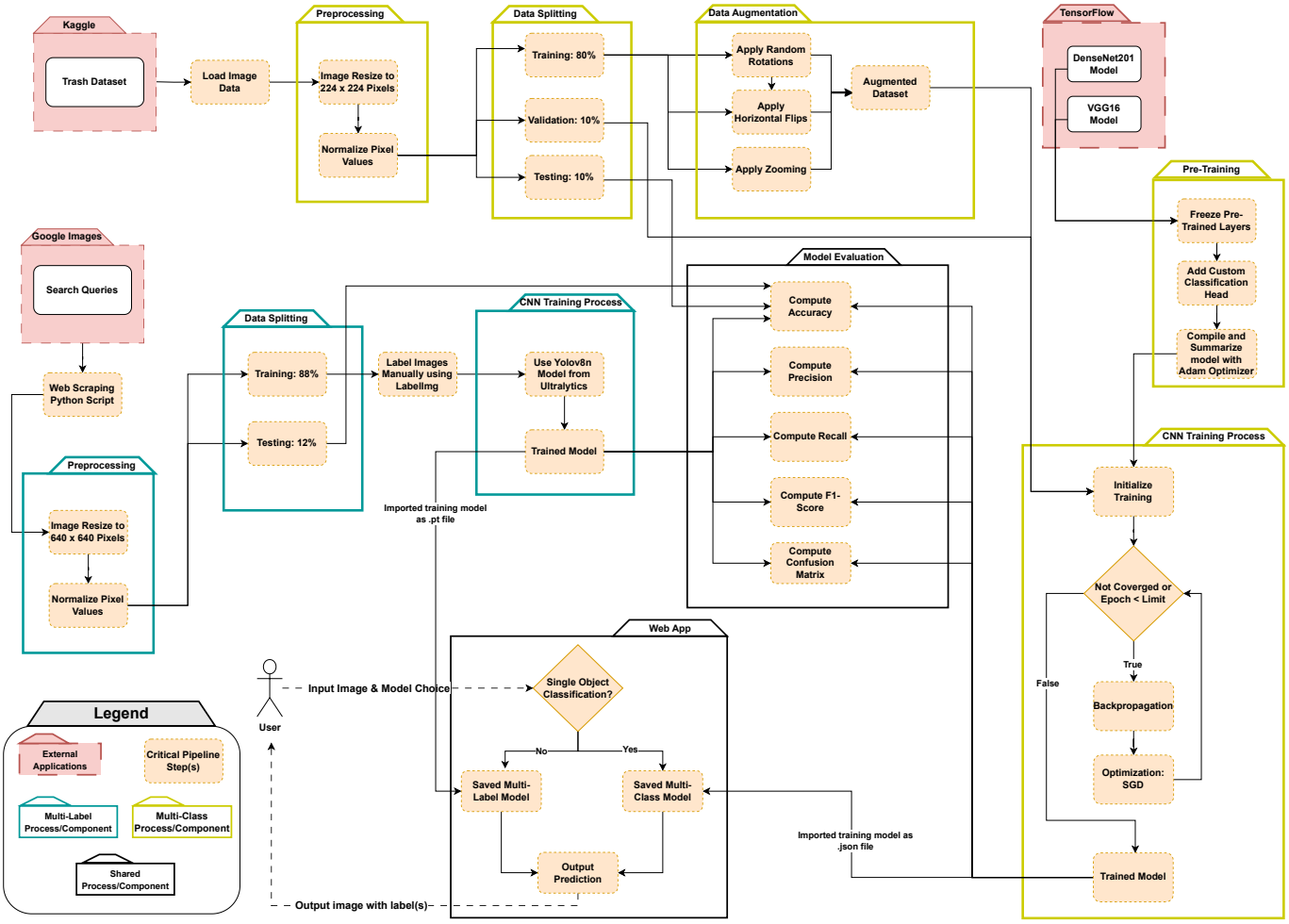


Figure 1: Model Training Pipeline

2.2 Dataset

The project uses a Trash Dataset obtained from Kaggle, which is directly uploaded to Google Colab's cloud-based platform [6]. This approach leverages the high-speed and powerful computational capabilities of Google Colab, facilitating the efficient handling of the dataset's large size [7]. Preprocessing the images involves resizing them to uniform 224x224 pixels, which is a necessary step to ensure consistency across all data inputs, a requirement for the neural network's architecture. Additionally, normalization of pixel values is carried out to mitigate the variances in lighting and color, thus aiding the model to converge more quickly and effectively during the learning process.

Data augmentation forms a significant part of the preprocessing phase and is implemented to enrich the dataset's variability. By introducing random rotations, horizontal flips, shearing, and zooming, the dataset is artificially expanded with different representations of the images. This variety is important in allowing the model to generalize across a larger set of image orientations and scenarios that mimic real-world conditions. These augmentations are essential since objects can be encountered in numerous positions and contexts. This ensures that the model is robust and capable of providing high predictive accuracy when deployed in practical applications.

Web-scraped images from Google, obtained using Selenium, are also integrated. These images are resized to 640x640 pixels and manually labeled using LabelImg due to the requirements of neural networks [2]. This process further diversifies the training set, ensuring the model's robust performance across various trash types and conditions. Detailed plots describing this custom dataset can be seen in Appendix Figure 3 and Appendix Figure 4.

2.3 Model Training

The model training regimen for the multi-class classification task employed the usage of the two advanced pre-trained models from TensorFlow's Keras library, DenseNet201 and VGG16. VGG16 is known for its depth and simplicity, consisting of 16 layers, predominantly convolutional, using small (3x3) filters in a uniform arrangement [8, 4, 9]. This model is particularly noted for its effectiveness in feature extraction due to its deep but straightforward architecture, making it a strong candidate for image classification tasks.

On the other hand, DenseNet201 is part of the Dense Convolutional Network (DenseNet) family and is characterized by its unique approach to connectivity [10, 3]. In DenseNet architectures, each layer is directly connected to every other layer in a feed-forward fashion. With 201 layers, DenseNet201 is efficient in terms of parameter usage and feature propagation within the network, due to its feature-reuse capability. This model stands out for its intricate layer connectivity, which enhances feature propagation and reduces the model's susceptibility to overfitting, making it highly efficient for complex image classification tasks [11, 12].

In the multi-class model, VGG16's architecture was leveraged to establish a baseline for performance. DenseNet201 was evaluated in parallel to determine its suitability for the specific requirements of the trash dataset and was ultimately selected over VGG16 for its superior accuracy and performance.

In the initial phase of training, the input image undergoes essential resizing to a uniform 224x224 pixels and normalizing the pixel values to ensure compatibility with the DenseNet201 architecture. The dataset is then divided into training, testing, and validation subsets in an 80-10-10 split, providing a balanced approach for training and performance evaluation, facilitated by the Sklearn, Pandas, and Numpy libraries. To further enhance the robustness of the model, data augmentation techniques like random rotations, horizontal flips, and zooming are applied. These techniques significantly increase the dataset's diversity and aid in the model's ability to generalize across different scenarios. The architecture is then augmented by adding a custom classification head to the pre-trained DenseNet201 base, featuring a global average pooling layer, multiple dense layers for learning higher-level abstractions, dropout layers to reduce overfitting, and a softmax activation layer in the output to compute class probabilities.

The architecture of the DenseNet201 model is enhanced for the specific task of image classification through the addition of a custom classification head. This head begins with a global average pooling layer, which serves the dual purpose of reducing the spatial dimensions of feature maps from the preceding layers while retaining the most critical spatial information. This reduction is crucial for decreasing the computational load and preventing overfitting by summarizing the features in a compact form.

The classification head also includes multiple dense layers, where each layer is activated by Rectified Linear Unit (ReLU) functions. ReLU is selected for its ability to introduce non-linearity into the model, enabling it to capture and learn complex relationships in the data. This non-linear activation function is particularly effective because it allows for faster training without significantly impacting the ability of the model to converge. In addition to dense layers, dropout layers with a rate of 0.3 are incorporated strategically within the classification head. These layers randomly deactivate a set proportion of neurons during each training iteration, effectively preventing any single neuron from becoming overly dominant in the learning process. This technique is a powerful tool against overfitting, as it encourages the network to distribute the learning across a wider set of neurons, leading to a more robust and generalizable model.

The final layer in the classification head is a softmax activation layer. This layer is crucial as it translates the outputs of the last dense layer into a probability distribution over the possible classes. The softmax function ensures that the output probabilities are normalized, making the model's predictions interpretable and comparable.

For compiling the model, the Adam optimizer is selected from the Keras library due to its effectiveness in handling sparse gradients and its adaptability in adjusting the learning rate during training. This optimizer combines the advantages of two other extensions of stochastic gradient descent, AdaGrad and RMSProp, making it well-suited for datasets with varying levels of complexity and size [13, 14, 15]. The loss function used is categorical cross-entropy, which is particularly appropriate for multi-class classification tasks as it quantifies the difference between the predicted probabilities and the actual distribution of the classes.

The training process is conducted iteratively, employing backpropagation for efficient computation of gradients and stochastic gradient descent (SGD) for updating the model's weights by leveraging the TensorFlow library. This combination ensures a balance between computational efficiency and the effectiveness of the learning process. Training is continued until the model achieves convergence, indicated by minimal improvements over iterations, or until a preset number of epochs is reached. This approach ensures that the model is adequately fine-tuned and capable of high

performance and reliability in classifying various types of waste materials, making it highly effective for practical applications in waste management and recycling. At the end of the model training, a json file is generated that outlines the architecture of the model, its layers, their order, the activation functions used, and other structural details. Additionally, an HDF5 file is produced which includes the trained parameters of the model. The resulting output of the application, once the input image is inserted into the trained model, will then appear as the original image with annotation of the class with the highest probability along with the probabilities of the other classes.

For the multi-label classification, the chosen YOLOv8n object detection model is a simplified version of the YOLOv8 model with a focus on speed and efficiency [16, 17, 18, 5]. To enhance information flow between layers, CSP (Cross-stage partial connections) are used by its backbone, a modified CSPDarknet53 network [19]. The C2f module effectively combines features from several layers, taking the place of the conventional YOLO neck architecture. Lastly, non-maximum suppression (NMS) and convolutional layers are used by a modified YOLOv5 head to precisely predict bounding boxes and class labels. To achieve impressive performance on edge devices, YOLOv8n uses a simplified yet efficient architecture. The reason for the selection of YOLOv8n among other variants of YOLOv8 are that it strikes an amazing balance between model size, accuracy, and speed. It is lightweight and efficient. On GPU's, YOLOv8n can infer quickly at up to 190 frames per second while maintaining a reasonable mAP of 50.7 on the COCO dataset. Its compatibility with ONNX, TensorFlow, and PyTorch makes it simple to implement across a range of platforms.

The multi-label training phase harnessed the YOLOv8n architecture via the Ultralytics Python package [5, 20]. The training and testing data split was done through the Roboflow environment and the ratio was determined through trial and error to be 88% and 12% respectively to optimize model performance. The dataset was uploaded to the Ultralytics hub, and training was initiated in a Jupyter notebook that accessed the dataset using an API key which enabled the effective use of cloud-based computational resources.

During the training phase, the model's accuracy was improved by data augmentation techniques, involving random rotations, crops, and flips to the images which was also conducted under the Roboflow website. Additionally, the model utilized anchor boxes from the annotated images to predict the position and dimensions of objects more accurately and employed non-maximum suppression which gets rid of overlapping duplicate bounding boxes. Upon completion of the training, a Pytorch file will be generated which contains the trained PyTorch model. Once the model is invoked with new unseen data, the output will then appear as the original image but with bounding boxes around the captured trash items that display the resulting class and its probability.

The overall length of the training period varied, potentially lasting several hours depending on the size of the dataset and the computational power available. It's important to note that under the hood, all the utilized libraries use SGD as a key algorithm for optimizing the learning process, thus this step is inherently abstracted from the actual code. After the training concluded, the model's accuracy and reliability were assessed using a separate validation set to ensure its readiness for deployment.

2.4 Prediction

After completing the training of the DenseNet201 model, the system is set up to handle the prediction of new images. The trained model, which was initially saved in JSON and HDF5 formats, is reloaded for use in prediction, thereby employing the same architecture and learned weights as established during the training phase. When a new image is introduced for classification, it is directly fed into the reloaded model without undergoing further preprocessing, as the model has already been optimized to handle such input through its initial training.

The DenseNet201 model, equipped with its deep convolutional layers, then proceeds to extract features from the input image. This feature extraction is a critical step, as the model leverages its training to identify patterns and characteristics specific to the different categories it has learned. The extracted features are passed through the model's architecture in a feedforward manner until the output layer is reached for the final prediction.

The model outputs the classification probabilities for each category, with the highest probability dictating the model's prediction. This prediction is not only a classification but also a measure of the model's confidence in its assessment. The result is then displayed alongside the image, offering a clear and understandable presentation of the model's prediction. Additionally, the system provides a practical interpretation of the item as either 'Recyclable' or 'Non-recyclable,' based on the predicted category. This approach ensures a straightforward and efficient process for predicting and understanding the classification of new images, demonstrating the practical application of the DenseNet201 model in image classification tasks.

For the multi-label classification process, a similar approach is adopted. After training, the YOLOv8n model is saved in a .pt file format, encapsulating its architecture and learned parameters. When predicting with the YOLO model, a new image is input into the reloaded system, where the model applies its learned feature detection capabilities to identify various objects. The YOLO model processes the image and propagates its features in a feedforward manner, using convolutional layers to detect objects and their respective categories, and outputs bounding boxes along with class labels along with the class probability for each detected object.

3 Results

3.1 Evaluation

In the multi-class classification model, the distribution of the dataset into six labels being used is shown in Appendix Figure 1. Understanding this data distribution is an important first step to help understand the classification system. One key observation of note is that the trash class has a much lower representation in the dataset. Once split into 80% train, 10% test & 10% validation, the distributions of the classes within the respective set can be determined. This is shown in Appendix Figure 2. The two models will be compared for performance on the distribution as a whole and as well on individual classes. The metrics being used to evaluate the models are: accuracy (on the training, test and validation sets), loss, precision, recall and F1 scores. The exact calculations and code for this evaluation procedure can be viewed in the Python notebook for the model.

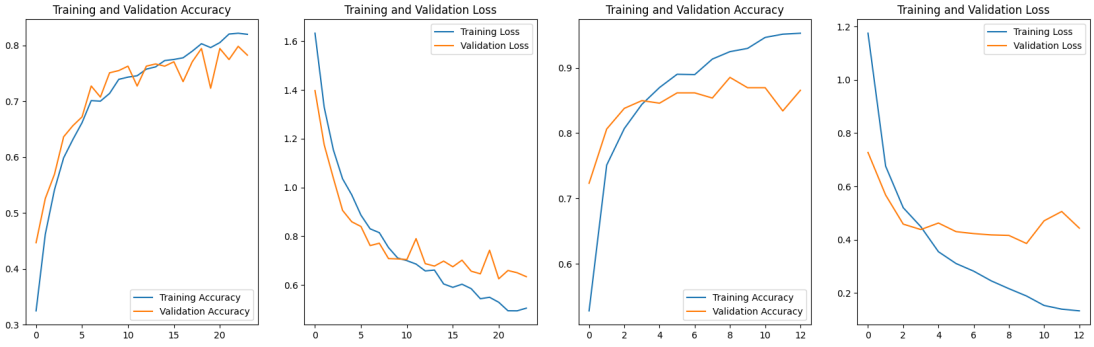
The dataset for multi-label is also divided into training and testing sets. This distribution for the multi-label dataset is illustrated in Appendix Figure 3. The YOLOv8n model's performance is evaluated on its ability to accurately detect and classify multiple objects using bounding boxes within images along with their attached probabilities and confidence. The evaluation metrics include the precision, recall, and F1 scores for each category, along with the overall accuracy and loss. Unlike the multi-class model, these metrics were generated in the Ultralytics Hub and were attached as screenshots in the final submission.

It is important to mention, that in YOLOv8's object detection, traditional testing and training accuracy metrics are less relevant because accurately determining true positives involves not just classifying an object correctly but also ensuring the predicted area closely matches the actual object's location. This dual requirement makes a straightforward accuracy calculation difficult since the object's position in the image becomes relevant. Metrics such as precision-recall curves and mAP scores are preferred for their ability to assess both detection accuracy and the precision of the predicted object regions. Moreover, F1-Score and support are similarly nuanced, as they must reflect the model's success in both detecting the correct number of objects (support) and balancing precision with recall (F1-Score).

There are no readily available models of a similar use case that are available to compare to. The baseline used to measure the model's performance is human ability. According to a recent survey on the population of the city of Saskatoon, about 47% of participants said a barrier to recycling was not knowing what can and cannot be recycled [21]. The range of accuracy in correctly classifying recyclable items from waste items varied drastically. Items such as paper products, cans and beverage containers were classified correctly in 88% or more of all trials. However, items such as aluminum products were classified correctly only 43% of the time [21]. These results demonstrate that random samples of residents in the community do have trouble with this classification task. The results shown ahead greatly exceed these baseline human performance metrics.

3.2 Results

For VGG16 and DenseNet201, the training duration was set to thirty epochs, with the flag with a directive to perform early stopping. VGG16 reached twenty-four out of thirty epochs before stopping. The accuracy and loss plots for both the training and validation sets are shown in Figure 2a. DenseNet201 reached thirteen epochs before stopping, requiring fewer epochs than VGG16 The accuracy and loss plots are shown in Figure 2b. DensenNet201 outperforms VGG16 in all but one metric. VGG16 shows slightly better precision for the classification of metal compared to DenseNet201. Table 1 shows the classification reports for both multi-class models. We can see the precision, recall and other metrics broken down by class. DenseNet201 outperforms VGG16 based on these metrics. Appendix Figure 5a-b shows the confusion matrices for both VGG16 and DenseNet201. Table 2 shows a summary comparison between the two models. DenseNet201 outperforms VGG16 in all aspects as it exceeds VGG16 performance while also requiring less training time. Finally, both models show higher accuracy on the training set than on the test set. Figures 5a-c show three samples of inferencing on new data. The results show there is a high degree of generalization.



(a) VGG16 (b) Densenet201

Figure 2: Accuracy & Loss values for Training & Validation Sets

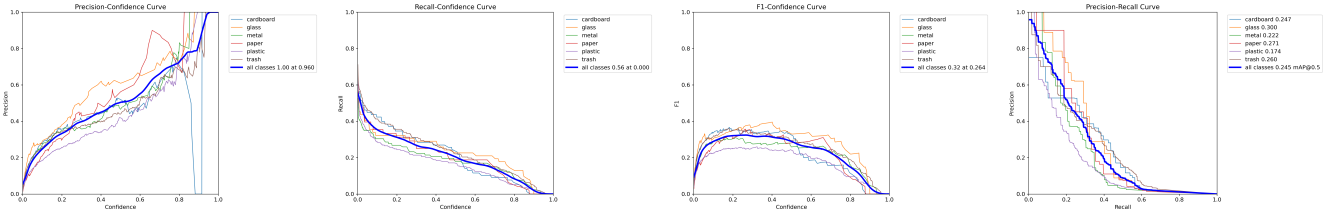
	VGG16				DenseNet101			
Class	**Precision**	**Recall**	**F1-Score**	**Support**	**Precision**	**Recall**	**F1-Score**	**Support**
cardboard	0.85	0.52	0.65	67	0.93	0.81	0.86	47
glass	0.76	0.75	0.75	51	0.88	0.85	0.86	52
metal	0.80	0.79	0.80	42	0.78	0.82	0.80	39
paper	0.78	0.80	0.79	59	0.92	0.92	0.92	60
plastic	0.50	0.86	0.63	28	0.88	0.91	0.89	46
trash	0.31	0.67	0.42	6	0.54	0.78	0.64	9
accuracy	-	-	0.72	253	-	-	0.86	253
macro avg	0.67	0.73	0.67	253	0.82	0.85	0.83	253
weighted avg	0.76	0.72	0.72	253	0.87	0.86	0.86	253

Table 1: Classification report for VGG16 & DenseNet201

Model	Training Accuracy	Testing Accuracy	Precision	Recall	F1
VGG16	81.7417	71.5415	66.8260	72.8937	67.3059
DenseNet201	96.9817	86.1660	81.9574	84.7111	82.8838

Table 2: Training and classification comparison between VGG16 & DenseNet201

The YOLOv8 training results, visualized through multiple performance curves in Figure 3, offer a comprehensive overview of the model's detection capabilities. The Precision-Confidence curve indicates that precision for all classes peaks at a perfect score at a high confidence threshold of 0.960, though this may not be fully representative of the model's general performance due to potential overfitting at such a high threshold. The Recall-Confidence curve shows that the recall for all classes decreases as the confidence threshold increases, which is typical as stricter confidence filters out more potential detections. The Precision-Recall curve further reveals that the model achieves a mAP50 of 0.245 across all classes, suggesting moderate overall precision and recall balance. Notably, the class-specific mAP50 values vary, with 'glass' achieving the highest score of 0.300, indicating a relatively better precision-recall balance for this class compared to others like 'plastic' with a lower mAP50 of 0.174. Lastly, the F1-Score-Confidence curve peaks at 0.32 for a confidence threshold of 0.264 for all classes, providing an operational point where the model achieves its best trade-off between precision and recall. These results are summarized in Table 3. Appendix Figure 5c-d shows a standard and a normalized confusion matrix for these classification results.



(a) Precision vs. Confidence (b) Recall vs. Confidence (c) F1-Score vs. Confidence (d) Precision vs. Recall

Figure 3: YOLOv8n results for precision, recall and F1-score

4 Discussion

4.1 Implications

The outcomes of this project indicate that the goal of creating an accurate waste classification application has been met. The model's high predictive accuracy and its adeptness at correctly identifying a broad spectrum of waste items are clear evidence of its success.

Key to achieving these results was the strategic use of the open-source DenseNet201 and YOLOV8n architecture, whose pretraining on complex image recognition tasks provided a solid foundation. This choice eliminated the need for an exhaustive search and testing of alternative neural network architectures, leading to significant savings in both time and computational effort.

The computational capabilities provided by Google Colab were another critical factor, enhancing the speed of model training and simplifying data management. The ability to harness such resources allowed for rapid iteration and development, which would have been considerably more challenging with limited computational power.

Class	Precision (Conf. 0.960)	Recall (Conf. 0.000)	F1-Score (Conf. 0.264)	Average Precision
Carboard	1.00	0.56	0.32	0.247
Glass	1.00	0.56	0.32	0.300
Metal	1.00	0.56	0.32	0.222
Paper	1.00	0.56	0.32	0.271
Plastic	1.00	0.56	0.32	0.174
Trash	1.00	0.56	0.32	0.260
All Classes	1.00	0.56	0.32	0.245 (mAP50)

Table 3: Classification report for YOLOv8n multi-label object detection

The quality of the dataset, with its accurate labelling and fair-quality images laid the groundwork for effective model training, which is often a stumbling block in machine learning projects due to poor data quality or insufficient labelling.

Furthermore, the implementation of data augmentation techniques enriched the training process, enabling the model to generalize its learning to new contexts. This was essential for the model's ability to recognize waste items in various orientations and settings, a critical feature for practical application in diverse environments.

With these aspects, the combined use of a robust pretrained neural network, powerful computational tools, a well-curated dataset, and sophisticated data augmentation strategies were crucial in achieving the desired outcomes of this waste classification project.

4.2 Strengths

The application showcases high predictive accuracy across a spectrum of waste categories, such as cardboard, glass, metal, paper, plastic, and general waste, with near-instantaneous classification results for images containing single and mixed waste items. Its user-friendly interface allows for easy image uploads, catering to a diverse user base and enhancing the ease of use. Scalability is a key design feature, with the web application ready to integrate future enhancements, including multi-label classification. The system's adaptability for use in various waste management systems, the minimal requirement for user expertise, and the capability to handle images of different qualities speak to its practicality in real-world settings. The strategic use of open-source technologies ensures that the application is both adaptable and maintainable, making it a robust tool in the domain of waste management.

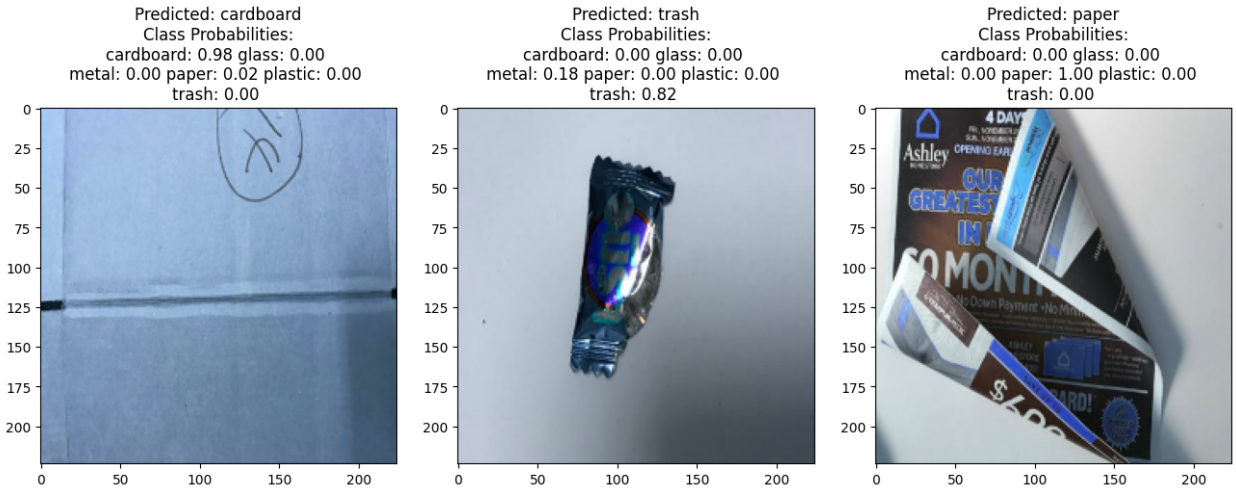
4.3 Limitations

The model faces challenges with non-standard inputs, like images with graphical overlays, text, or symbols, often attempting incorrect classifications. This limitation extends to digitally altered images or atypical content, such as a waste item photograph with added text. Its classification capability is limited to the predefined dataset and labels used in training, thus not encompassing certain waste types like electronic or hazardous waste. Additionally, the model lacks functionality to advise on the disposal of non-recyclable items.

4.4 Future Directions

Expanding the scope of the model is a key future goal. This expansion will encompass a broader range of waste categories and will be trained on a more realistic collection of images for the training set. Currently, the training set consists of a mixture of real-world images along with web-scraped images. These web-scraped pictures may be heavily edited thus affecting the model's accuracy. Future enhancements to the model should also address the current imbalance in the dataset, where certain waste categories are over-represented, potentially biasing the classification process. Equally distributing the number of images across all categories is essential to prevent skewness that may affect the model's accuracy, especially in distinguishing between visually similar such as glass and plastic. The model may confuse these items as they share similar characteristics.

To enhance the accuracy of the models, focusing on curating a realistic image set through targeted field data collection and crowdsourcing should be explored. This data collection can include partnering with local waste management facilities to collect a diverse range of real-world images of various waste types, organizing community clean-up events to photograph different types of waste in natural settings, and developing a mobile app for users to contribute waste item pictures. Additionally, collaborating with educational institutions for waste classification projects and launching social media challenges for people to share images can further enrich the dataset with a wide array of waste items in different states. Furthermore, the group should implement algorithms to filter out heavily edited web-scraped images along with developing specialized sub-models focused on identifying unique material characteristics such as reflectivity, transparency, and texture. These sub-models, trained on dedicated datasets comprising a diverse range of images under varied conditions, will enhance the model's ability to discern subtle differences between challenging categories.



(a) Cardboard (b) Trash (c) Paper

(d) Multi-label Example 1 (e) Multi-label Example 2

Figure 4: Example predications with class probabilities.

5 Additional Questions

5.1 What are the feedbacks that you found useful from the peer evaluation?

Among the feedback provided, three particular suggestions were found to be especially useful. The first feedback recommends that the introduction should directly state what the application does, avoiding early mention of the methodologies to keep the initial explanation straightforward. The second feedback calls for a more detailed explanation in the results section about the specific aspects that contributed to the predicted class, giving users insight into the decision-making process. The third feedback encourages the use of actual examples or scenarios to show the application's real-world benefits, helping users see its effectiveness in practical waste management scenarios. These points collectively seek to make the report more reader-friendly and informative.

5.2 What changes did you make based on the feedback from peer evaluation?

The introduction was revised to align with the first feedback, ensuring the application's purpose is immediately clear. Technical details were deferred to the methodology section for greater clarity. The second feedback, regarding the explication of neural network predictions, wasn't acted upon. Neural networks, due to their complex nature, don't readily offer the transparency needed to describe the influence of individual features on the predictions. In lieu of a complete explanation, the project has accommodated this feedback by including the probability of alternative classes in the predictions, offering a measure of interpretability within the confines of the model's complexity. The third feedback suggested including a discussion on real-world scenarios, but due to the report's page limit, the focus should remain more on discussing the methodology as the intent of this project is to showcase what was learned in the course. In a future version with less restrictive constraints, the inclusion of practical examples to demonstrate the model's utility would be a priority.

References

[1] "Trashbot: The smart recycling bin that sorts at the point of disposal." https://cleanrobotics.com/trashbot/, Oct 2023.

[2] "labelimg." https://github.com/HumanSignal/labelImg?ref=blog.roboflow.com.

[3] "Densenet201." https://www.tensorflow.org/api_docs/python/tf/keras/applications/densenet/DenseNet201.

[4] "Vgg16." https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16.

[5] "Ultralytics yolov8 docs." https://docs.ultralytics.com/.

[6] CCHANG, "Garbage classification." https://www.kaggle.com/ds/81794, 2018.

[7] "Google colaboratory." https://colab.google/, 2023.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." https://arxiv.org/abs/1409.1556, 2015.

[9] R. Thakur, "Beginner's guide to vgg16 implementation in keras." https://builtin.com/machine-learning/vgg16, March 2023.

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." https://arxiv.org/abs/1608.06993, 2018.

[11] Z. Aslan, "Transfer learning using densenet201." https://www.yesilscience.com/transfer-learning-densenet201/.

[12] fchollet, "Developer guides: Transfer learning fine-tuning." https://keras.io/guides/transfer_learning/, June 2023.

[13] "Adam." https://keras.io/api/optimizers/adam/.

[14] "Adagrad." https://keras.io/api/optimizers/adagrad/.

[15] "Rmsprop." https://keras.io/api/optimizers/rmsprop/.

[16] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 and beyond." https://arxiv.org/abs/2304.00501, 2023.

[17] A. Mehra, "Understanding yolov8 architecture, applications features." https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/, April 2023.

[18] S. Rath, "Understanding yolov8 architecture, applications features." https://learnopencv.com/ultralytics-yolov8/, January 2023.

[19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection." https://arxiv.org/abs/2004.10934, 2020.

[20] "ultralytics." https://github.com/ultralytics/ultralytics.

[21] "City of saskatoon 2021 waste recycling survey." https://www.saskatoon.ca/sites/default/files/documents/cos_2021_waste_recycling_survey.pdf, 2021.

A Appendix: Supplementary Figures

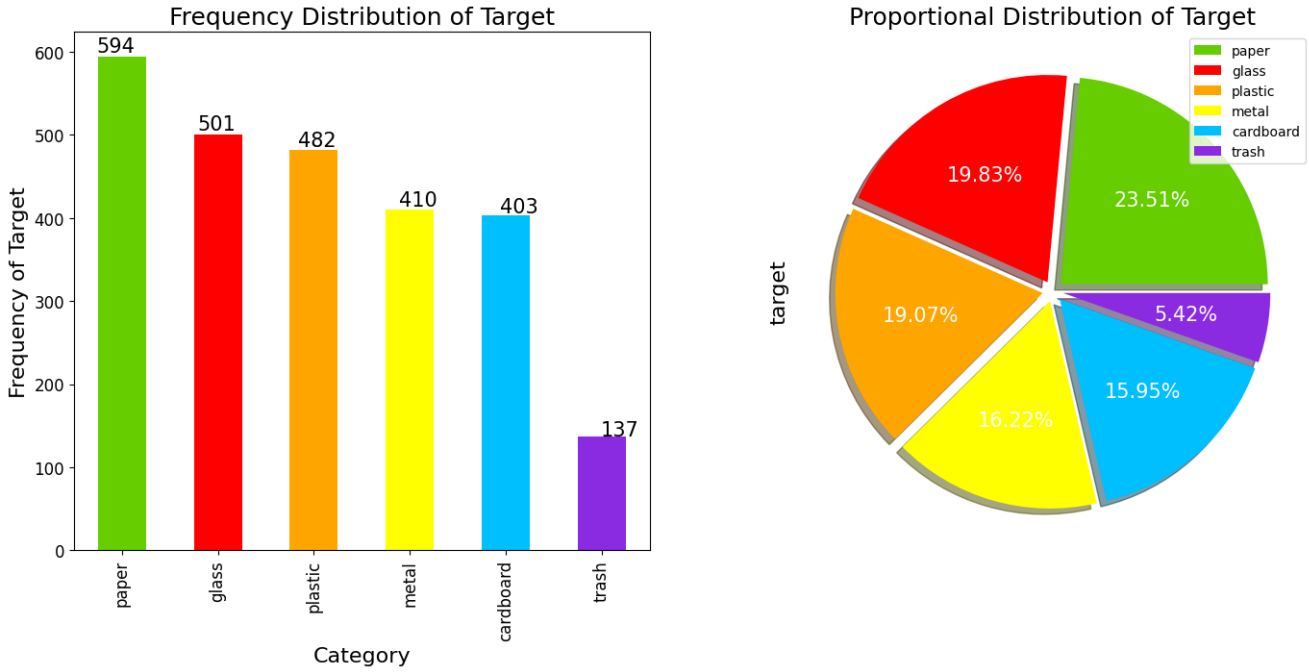


Figure 1: Distribution of labels in the dataset

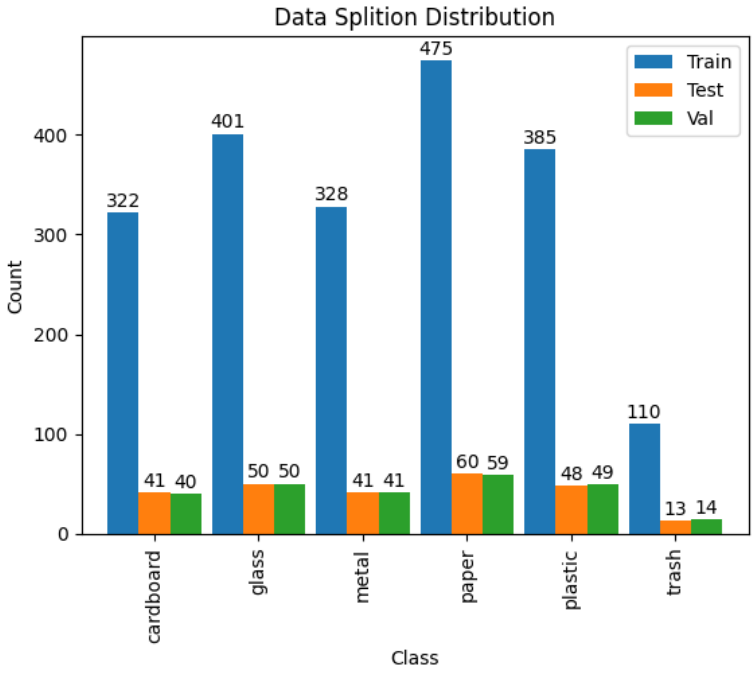


Figure 2: Splitting of dataset into Training Set, Test Set & Validation Set

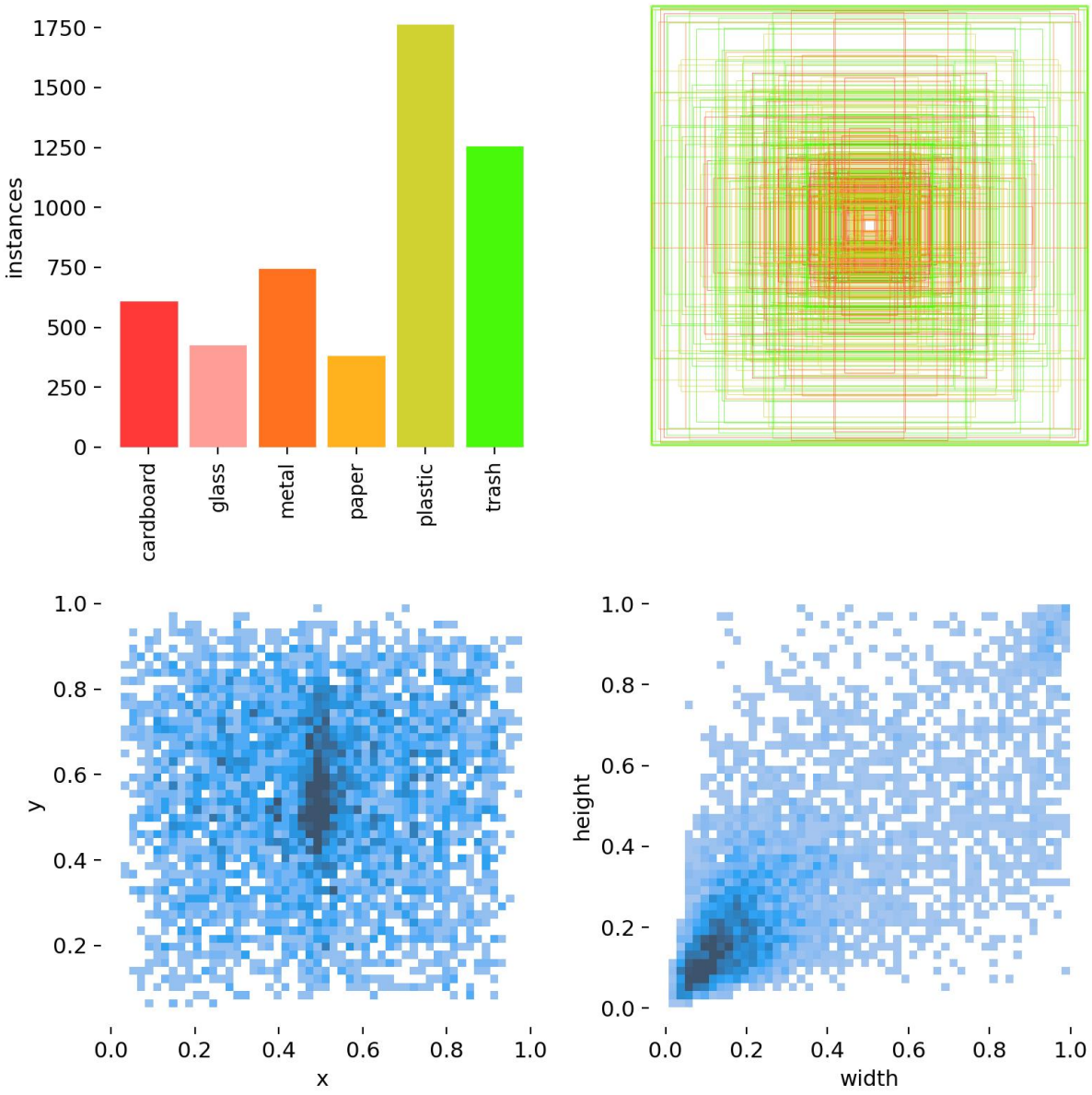


Figure 3: Multi-label classification dataset label distribution

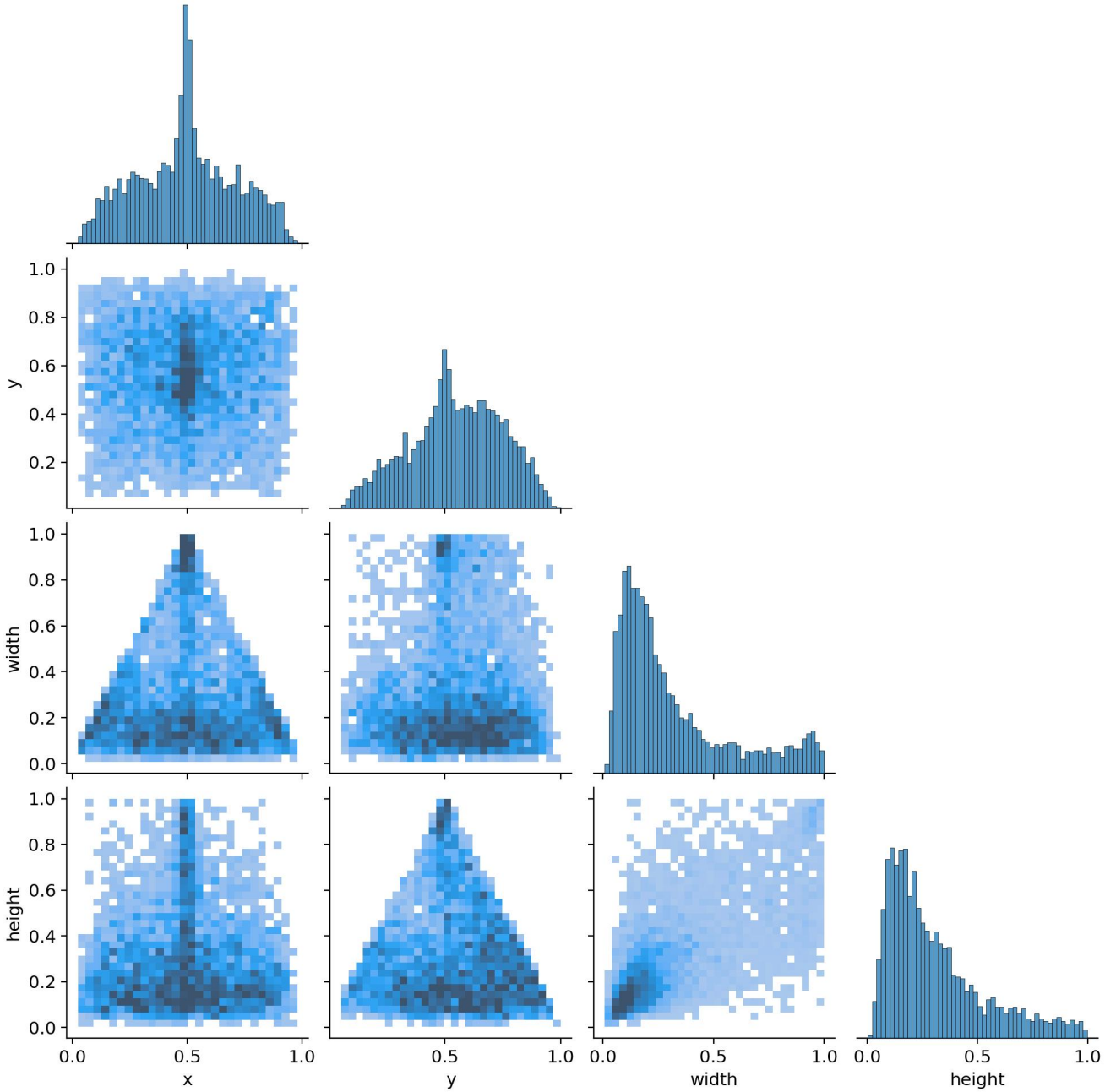
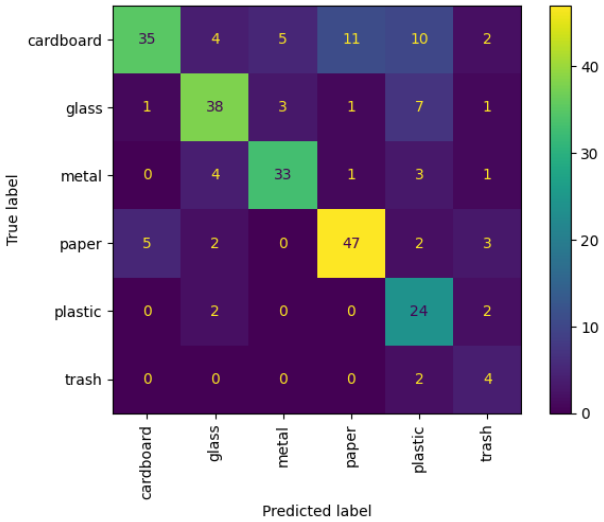
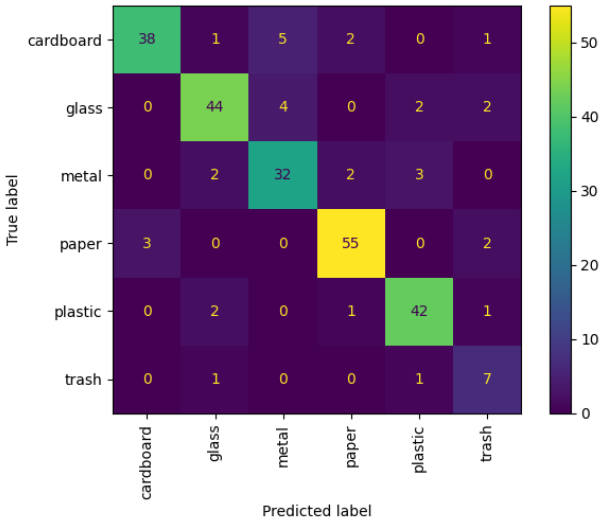


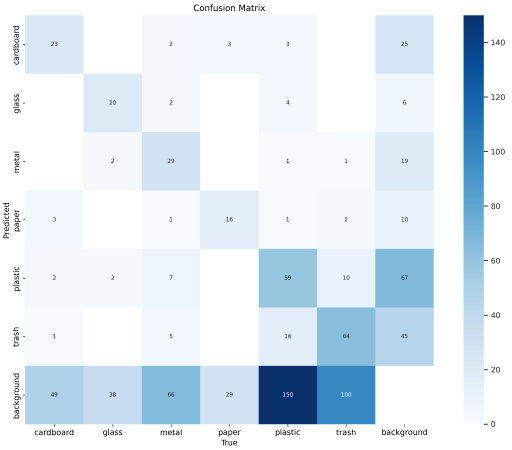
Figure 4: Multi-label classification dataset label correlogram



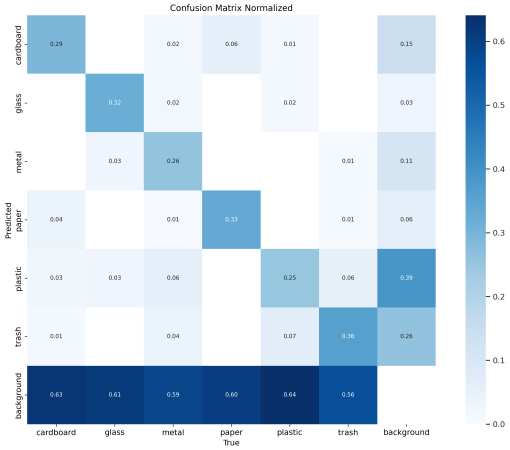
(a) VGG16



(b) Densenet201



(c) YOLOv8n



(d) YOLOv8n Normalized

Figure 5: Confusion Matrices