YOLO-Based Real-Time Waste Classification with Dual Input

Francheska L. Olympia
College of Computing
and Information Technologies
National University - Manila
Manila, Philippines
olympiafl@students.national-u.edu.ph

Renz Andrei C. Alis
College of Computing
and Information Technologies
National University - Manila
Manila, Philippines
alisrc@students.national-u.edu.ph

Denmar Yzelle V. Saralde
College of Computing
and Information Technologies
National University - Manila
Manila, Philippines
saraldedv@students.national-u.edu.ph

Charles Angelo R. Racho
College of Computing
and Information Technologies
National University - Manila
Manila, Philippines
rachocr@students.national-u.edu.ph

Xavier Pagdanganan
College of Computing
and Information Technologies
National University - Manila
Manila, Philippines
pagdangananx@students.national-u.edu.ph

***Abstract*—Effective waste classification is critical for promoting sustainable waste management and improving recycling efficiency. Manual sorting methods are often labor-intensive and prone to error, underscoring the need for automated solutions. This study presents a real-time object detection model based on a YOLO-inspired Convolutional Neural Network (CNN) architecture for classifying waste into five categories: glass, metal, paper, plastic, and leaf. Multiple model architectures were evaluated, and the final model achieved a classification accuracy of 94%. The system demonstrates strong potential for deployment in automated waste segregation setups, supporting more efficient and environmentally friendly waste processing**

***Index Terms*—Waste Classification, Deep Learning, CNN, YOLO, ResNet-50, Real-time Detection, Dataset Augmentation**

I. Introduction

Improper waste disposal has long been a major issue in both rural and urban areas. The mismanagement of waste results to numerous negative effects on communities, including environmental pollution, health hazards, and inefficient recycling practices that can even lead to flooding even with just light rains. In the Philippines, it is estimated that around 35,000 tons of garbage are generated daily [1], with most of the waste consists of plastics, paper, and kitchen wastes [2]. In addition to this, 81% of the plastic garbage was thrown into the ocean [3] further worsening the pollution in water environments and endangering the aquatic life. Due to the rapid industrialization and urbanization, it causes an extraordinary increase in the origination of unwanted waste, and makes segregation a problem for the community even though there are a lot of programs implemented for waste management such as separating waste from a recyclable one [4].

Today, waste management is a very common term and is used to describe series of activities from waste generation to disposal that could help sort the innumerable problems due to improper waste disposal that includes the adverse effects on the human health and the environment [5]. Despite there are existing studies and implementations of waste-related policies, there are still many communities in the country that continue to struggle with the consistent and efficient waste disposal. Local government units (LGUs) often face limitations when it comes to their resources, infrastructure, and community participation. Accordingly, this existing gap between policy and the actual implementation shows the need for more a more innovative and accessible approaches that addresses both the structural and behavioral challenges

The growing global waste problem poses serious risks to environmental sustainability, human health, and the availability of natural resources. Traditional waste management methods, which often rely on manual sorting, are inefficient and prone to errors. This leads to contamination of recyclable materials and an increased reliance on landfills. To address these issues, there is a need for intelligent and automated systems that can accurately classify waste. This paper presents YOLO-Based Real-Time Waste Classification with Dual Input. The main objective of this project is to create a machine learning model that identifies the waste provided through webcam or uploading an image and then classifies it into biodegradable, nonbiodegradable, and recyclable categories to help improve sorting accuracy, reduce human workload, and support more effective and responsible waste management practices in the real world.

The significance of this project is that it tends to enhance the effectiveness of waste management to provide proper garbage organization to lessen the improper disposal of trashes and can contribute to a more sustainable and environment for all with the help of the machine learning model created.

II. Literature Review

In the field of computer vision, object detection remains a crucial part for applications that use automatic surveillance systems. As it now becomes an in demand, a good real-time with a high-accuracy detection is a must. Researchers and Machine Learning models continues to develop various deep learning architectures to provide an effective model that is efficient for everyday use.

A. Convolutional Neural Networks (CNN)

In deep learning, the foundational architecture is Convolutional Neural Network or also known as CNN, wherein it particularly excels in image recognition, classification, and object detections. This model was introduced by LeCun et al. in the 1990s for digit recognition. CNN became popular because of the success of AlexNet in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in the year 2012, where it exemplifies the potential of deep learning models to outperform the traditional image processing methods [6]. CNNs purpose is to automatically adapt and learn spatial hierarchies of features through backpropagation by using a combination of convolutional, pooling, and fully connected layers. This kind of structure made CNN to be effective on extracting local features from images while maintaining a spatial relationship. As time have passed, there came numerous enhancements on CNN architectures such as VGGNet, GoogLeNet, ResNet and DenseNet, these enhanced architectures moving over from the boundaries of computational efficiency and accuracy [7]. The continued development of CNNs has enabled researchers to create models that are not only accurate but also optimized for deployment on resource-constrained devices through techniques such as model compression, quantization, and lightweight network design [8].

B. You Only Look Once (YOLO)

Before, object detection systems repurpose the classifiers to perform detection. In detecting an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. A model introduced by Redmon et al in the year 2016 shows a better and faster object detection which is named You Only Look Once (YOLO). [9]. Unlike the traditional detection models, YOLO treats object detection as a single regression problem. It simultaneously predicts bounding boxes and class probabilities directly from the full images using a single convolutional neural network (CNN). This innovated architecture based from CNN enables the new YOLO model to achieve a remarkable speed, making it reliable to real-time applications because it is faster than the older model.

There are YOLO models that have incorporated pretrained backbone networks to improve its features and detection performance without worrying about its efficiency and building from scratch. For instance, YOLOv3 employs a Darknet-53, a deeper convolutional network that was pretrained on ImageNet as its backbone [10]. With the use of pretrained models, it allows models like YOLO to leverage learned representations from large datasets, enhancing its ability to detect objects across diverse classes and challenging datasets and environments. These improvements made by pretrained models have contributed to the widespread of YOLO's adoption in the real world applications that requires fast and accurate detection.

C. Understanding Training Loss, Accuracy and Validation Loss, Accuracy

Evaluating the model is important to know and understand the learning process and generalization of it. Training Loss measures the error of the model inside the training dataset and usually it decreases as the model learns a new pattern within the dataset. On the other hand, Validation loss evaluates the model's performance on unseen or new data to help detect overfitting and underfitting during the training of the deep learning model [11]. For the training accuracy, it reflects how well the model could predict the labels for the training data provided. While the validation accuracy measures the predictive accuracy on the validation data, providing the insights into how the model might perform in real-world scenarios [12]. The separation between training and validation metrics can be indicated to potential issues such as if the training accuracy continues to improve while the validation accuracy decreases, the model is most likely to be overfitting the training data [13].

If the training and the validation loss remains high, this means the model may be underfitting, indicating insufficient learning from the dataset. Understanding and analyzing these metrics carefully throughout the training could help the machine learning engineer or anyone working with the model decide when to stop the training (early stopping), adjust the hyperparameters, or to modify the model complexity to proceed to a better model generalization and robustness [14].

D. Understanding CNN architecture

Convolutional Neural Network, or also known as CNN have developed significantly, each bringing their own innovations that has improved in model accuracy, computational efficiency, and feature extraction capabilities. Early CNNs like LeNet-5 used a foundation with shallow layers that is suitable for basic patter recognition [15]. Next was AlexNet, wherein it was introduced with a deeper network with the ReLU activation and dropout regularization, resulting to a remarkable performance on the ImageNet dataset and leading to a modern era of deep learning architectures [16]. Eventually, VGGNet model emphasized an architecture of depth with uniform 3x3 convolutions, trading off the model size for improved accuracy, while GoogleNet introduced a multi-branch architecture to process the information at multiple scales in parallel which improves efficiency without signifcantly increasing the parameters [17].

ResNet addressed degredation issues in deep networks by introducing residual connections, enabling the training of models that contains hundreds of layers without vanishing its gradients [18]. On the other hand, lightweight models such as MobileNet and EfficientNet have gained the popularity for deployment on limited-resource devices due to their effectivity respectively [19]. Each architecture offers trade-offs between their accuracy, speed, and memory consumption allowing the choosing of CNN model architecture dependent on the system application and specifications. Understanding such architectures is critical not only for optimization of the performance but also to ensure the feasibility of the deployment of deep learning models to real-world systems like mobile vision, medical diagnostics, and real-time detection systems [20].

III. Methodology

A. Dataset

The dataset utilized in this study was meticulously compiled from multiple sources to ensure a diverse representation of waste materials. The primary component of our dataset originated from TrashNet, a publicly available dataset by Feyza Ozkefe, last updated approximately four years ago on Kaggle. This initial TrashNet dataset was subsequently uploaded to Roboflow for comprehensive annotation and integration.

To expand and diversify the dataset, additional images were sourced from various readily available public datasets on Roboflow. Some of these supplementary datasets were pre-annotated, while others required manual annotation. The selection of these additional datasets aimed to augment the class representation, with images randomly selected and incorporated to achieve a threshold of approximately 1,000 images per class. This process involved an iterative approach of adding around 200 images at a time until the desired class balance was achieved.

The final dataset is structured into five distinct classes: glass, leaf, metal, paper, and plastic.

- Glass: Includes bottles, clear glasses, jars, and broken glass fragments.
- Leaf: Contains diverse leaves, predominantly green, with some brown or wilted specimens.
- Metal: Comprises soda cans, canned goods, aluminum foil, aluminum containers, and metal lids.
- Paper: Features white paper, magazines, crumpled paper, food cartons, cardboard, and boxes.
- Plastic: Encompasses plastic bags, plastic packaging (e.g., snack wrappers, coffee, candy bags), plastic bottles, and small plastic containers.

Classes	**Image Count**	**Annotation Count**
Glass	900	1101
Leaf	769	1006
Metal	1009	1034
Paper	1071	1074
Plastic	996	1193

Fig. 1. Dataset Summary

B. Preprocessing

After collecting and initially annotating the dataset in Roboflow, it was subjected to several pre-processing steps to standardize and prepare the images for model training. The images underwent auto-orientation to correct any accidental rotations, ensuring all images were uniformly oriented and preventing potential confusion for the model. Next, all images were resized to a target dimension of 640×640 pixels, a common recommended size for object detection. To preserve the original aspect ratio and prevent distortion, the resizing was configured to fit images within black padded edges.

To further enhance the dataset's diversity and robustness, various image augmentation techniques were applied within Roboflow. This involved generating synthetic variations of the original images, which included:

Flipping: Both vertical and horizontal flips were applied. Rotation: Random rotations between -15 degrees and +15 degrees were implemented. Light Condition Simulation: Adjustments to brightness (-15% to +15%), saturation (-25% to +25%), and exposure (-10% to +10%) were performed to simulate varying lighting environments.

These augmentations resulted in a total of 12,639 images. Roboflow's built-in feature was then used to split the dataset directly into training, validation, and testing sets, ensuring a standardized partition. The final distribution was approximately 90.84% for training (11,481 images), 5.10% for validation (645 images), and 4.06% for testing (513 images).

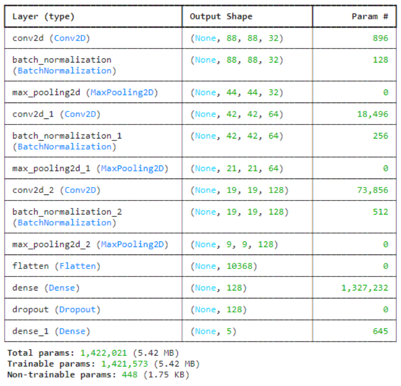
For model training, the pre-processed and split dataset was fetched using the Roboflow API in YOLOv8 format. Upon downloading and extracting the data in the Google Colab environment, the necessary file directory was set up. The provided data.yaml file was utilized to define the class names, and subdirectories were created for each image class, sorting the images with their corresponding labels. Finally, for broader classification and mapping, these classes were categorized based on their biodegradability and recyclability status using the following scheme:

- Glass: non-biodegradable, recyclable
- Leaf: biodegradable, non-recyclable
- Metal: non-biodegradable, recyclable
- Paper: biodegradable, recyclable
- Plastic: non-biodegradable, recyclable

C. Model Architecture

In determining the best model for the project, multiple CNN architectures were designed and evaluated, each varied in convolutional depths and layer configurations. To identify the most optimal architecture for waste classification and detection, four different machine learning models were designed and evaluated:

1) Baseline CNN Model: A traditional convolutional neural network was first implemented to serve as the basis. It consisted of three convolutional blocks with batch normalization and max-pooling layers, followed by fully connected layers for classification. This model's focus was on classification only and did not perform object localization.

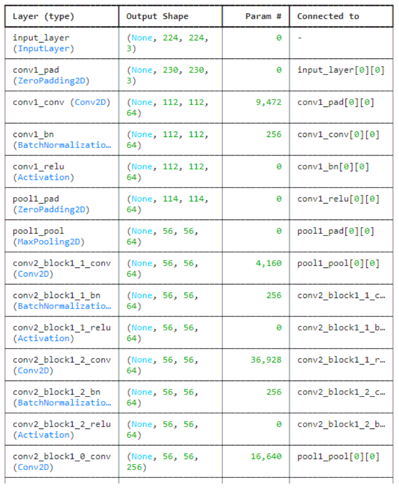


Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 88, 88, 32)	896
batch_normalization (BatchNormalization)	(None, 88, 88, 32)	128
max_pooling2d (MaxPooling2D)	(None, 44, 44, 32)	0
conv2d_1 (Conv2D)	(None, 42, 42, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 42, 42, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 21, 21, 64)	0
conv2d_2 (Conv2D)	(None, 19, 19, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 19, 19, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 9, 9, 128)	0
flatten (Flatten)	(None, 10368)	0
dense (Dense)	(None, 128)	1,327,232
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 5)	645

```
Total params: 1,422,021 (5.42 MB)
Trainable params: 1,421,573 (5.42 MB)
Non-trainable params: 448 (1.75 KB)
```

Fig. 2. Baseline CNN model Architecture

2) *ResNet-50 with Transfer Learning Model:* A more complex model was created using ResNet50 as its backbone, pretrained on ImageNet. The final layers of the model were customized with a global average pooling and fully connected layers. The last 50 layers of ResNet50 were fine-tuned while freezing the remaining layers to retain the learned features. This model was limited to classification only.

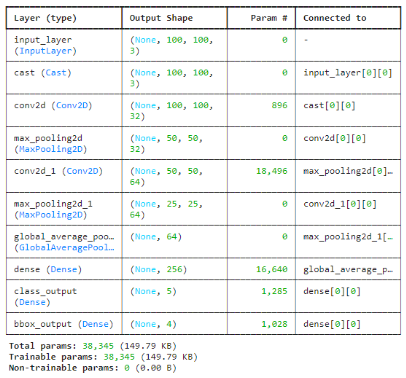


Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_b…
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_r…
conv2_block1_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_2_c…
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_b…
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,640	pool1_pool[0][0]

Fig. 3. A Snippet of ResNet-50 with Transfer Learning Model Architecture

3) *Trial 1 of YOLO-inspired CNN Model:* A YOLO-like architecture was made from scratch by using two convolutional blocks and dual outputs: one for class prediction (softmax) and one for bounding box regression (sigmoid). This demonstrated the feasibility of simultaneous classification and localization of the model but lacks depth for complex feature extractions.

4) *Final YOLO-inspired Model:* Building upon the insights from the first YOLO-like trial, a deeper and more robust YOLO-inspired architecture was developed to perform both classification and object localization. This final model consists of four convolutional layers with increasing filter sizes, each followed by max-pooling layers to reduce dimensionality while preserving spatial features. A Global Average Pooling (GAP) layer is used to condense spatial information into a compact representation, followed by a dense layer with 256 units and ReLU activation.



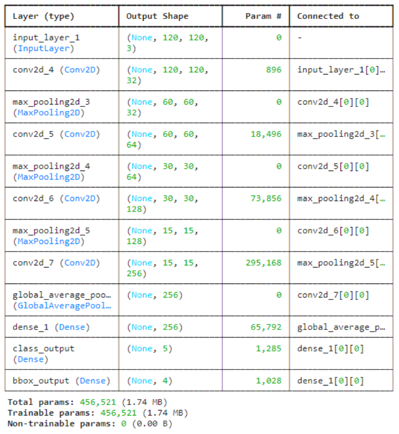
Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 100, 100, 3)	0	-
cast (Cast)	(None, 100, 100, 3)	0	input_layer[0][0]
conv2d (Conv2D)	(None, 100, 100, 32)	896	cast[0][0]
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 50, 50, 64)	18,496	max_pooling2d[0]…
max_pooling2d_1 (MaxPooling2D)	(None, 25, 25, 64)	0	conv2d_1[0][0]
global_average_poo… (GlobalAveragePool…	(None, 64)	0	max_pooling2d_1[…
dense (Dense)	(None, 256)	16,640	global_average_p…
class_output (Dense)	(None, 5)	1,285	dense[0][0]
bbox_output (Dense)	(None, 4)	1,028	dense[0][0]

```
Total params: 38,345 (149.79 KB)
Trainable params: 38,345 (149.79 KB)
Non-trainable params: 0 (0.00 B)
```

Fig. 4. Trial 1 YOLO-inspired Model Architecture

The model features two output branches: class_output for multi-class classification using a softmax activation function, and bbox_output for bounding box prediction using sigmoid activation to output normalized (x, y, width, height) values. A multi-task learning approach was applied that the classification head is trained using categorical cross-entropy, and the bounding box head is optimized using mean squared error (MSE). The model is compiled using the Adam optimizer with a learning rate of 0.001.

This YOLO-like architecture, implemented entirely from scratch CNN, balances performance and efficiency, making it suitable for real-time waste classification scenarios. Among all the models tested, this final YOLO-inspired model achieved the highest accuracy and generalization, and was therefore selected as the final architecture for deployment and detailed evaluation.



Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 120, 120, 3)	0	-
conv2d_4 (Conv2D)	(None, 120, 120, 32)	896	input_layer_1[0]…
max_pooling2d_3 (MaxPooling2D)	(None, 60, 60, 32)	0	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 60, 60, 64)	18,496	max_pooling2d_3[…
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 64)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 30, 30, 128)	73,856	max_pooling2d_4[…
max_pooling2d_5 (MaxPooling2D)	(None, 15, 15, 128)	0	conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 15, 15, 256)	295,168	max_pooling2d_5[…
global_average_poo… (GlobalAveragePool…	(None, 256)	0	conv2d_7[0][0]
dense_1 (Dense)	(None, 256)	65,792	global_average_p…
class_output (Dense)	(None, 5)	1,285	dense_1[0][0]
bbox_output (Dense)	(None, 4)	1,028	dense_1[0][0]

```
Total params: 456,521 (1.74 MB)
Trainable params: 456,521 (1.74 MB)
Non-trainable params: 0 (0.00 B)
```

Fig. 5. Final YOLO-inspired Model Architecture

D. Model Training

Each model was trained with standardized procedures, tailored to the specific characteristics and computational requirements of the respective networks. Image sizes, batch sizes, callbacks, and the epochs were all adjusted accordingly from experiments and memory constraints of the hardware specifications used.

1) Baseline CNN Model: The simple CNN architecture was trained with 50 epochs using a smaller batch of size 8 to accommodate the memory limitations of the hardware. The training contained EarlyStopping to prevent overfitting and ReduceROnPlateau to adjust the learning rates as the training progressed.

2) ResNet-50 with Transfer Learning Model: The ResNet model was trained with only 8 epochs with three callbacks such as EarlyStopping to halt the training when validation loss stopped improving with a patience equal to 5. ReduceROnPlateau to lower its learning rate dynamically upon performance with a patience equal to 2. And lastly ModelCheckpoint to retain the best performing model based on the validation accuracy.

3) Trial 1 of YOLO-inspired CNN Model: The first YOLO-like model was trained with 10 epochs using resized image inputs at 100x100 pixels and a batch size of 32. This configuration maintained a balance between the feature retention and preventing out-of-memory errors. The training included: TensorBoard, for real-time tracking of the metrics. ModelCheckpoint, for saving the best model made. And EarlyStopping, to prevent overfitting of models.

4) Final YOLO-inspired Model: The last and refined model was trained over 50 epochs with an image size of 120x120 pixels and a batch size of 8 to prevent RAM crash. A more thorough training flow was applied: TensorBoard, for a detailed visualization. ModelCheckpoint, for saving the best model. EarlyStopping, for regularization or prevention of overfitting. And traininglogger, to track the accuracy and loss per epoch.

E. Model Evaluation

To assess the performance of the implemented models, the four different architectures undergo training and was evaluated using accuracy, loss trends, and F1 Scores.

1) Baseline CNN Model: The baseline CNN model trained with 10 epochs went through overfitting with the fifth epoch, initially, it achieved a training accuracy of 50.6% and validation accuracy of 60.2%, reaching about 67.5% training accuracy. The model served as a benchmark for subsequent experimentation on model architectures.

2) ResNet-50 with Transfer Learning Model: While ResNet was expected to outperform the models due to its deeper architecture and pretrained wights, it encountered issues with data exhaustion during the training process. The model managed to achieve a 56.8% training accuracy by the second epoch, but the validation accuracy remained at 0, and the validation loss spiked to values like 72.67 and 11.18. This suggests to potential issues in data handling, and also insufficient system memory which made it unreliable for final evaluation and was not selected for deployment.

3) Trial 1 of YOLO-inspired CNN Model: The first trial of YOLO inspired architecture showed superior performance over the first two model, with 10 epochs used, the training accuracy improve from 32.0% to 72.0%, while the valdiation accuracy increased to 71.4%. This model also reported a consistent decline in classification loss and total loss, alongside with a stable bounding box regression loss. The overall accuracy reached 73.54% reflecting to a strong multi-class classification performance, especially on leaf and metal categories. The custom model proved effective in jointly learning object localization and classification, offering a practical balance between accuracy and computational efficiency.

4) Final YOLO-inspired Model: The final YOLO-inspired model was trained over 50 epochs, with each epoch taking approximately 5.9 minutes on average, leading to a total training time of approximately 4.9hours. The classification accuracy improved steadily across epochs from 42.68% at the beginning to 96.16% on the training set, and a corresponding validation classification accuracy of 94.07%, indicating minimal overfitting and strong generalization. Overall, this YOLO-based CNN architecture, trained from scratch without the use of pretrained models, proved highly effective for real-time waste detection and classification, offering both high accuracy and reliable bounding box predictions across diverse categories.

IV. Results and Discussion

The training performance of our model, monitored across 29 epochs, provided clear insights into its learning and generalization. We see the training loss consistently drop from 1.3601 at Epoch 1 to 0.1305 by Epoch 29. Similarly, training accuracy (class_output_accuracy) steadily climbed from 0.4268 to 0.9596, showing strong learning on the training data.

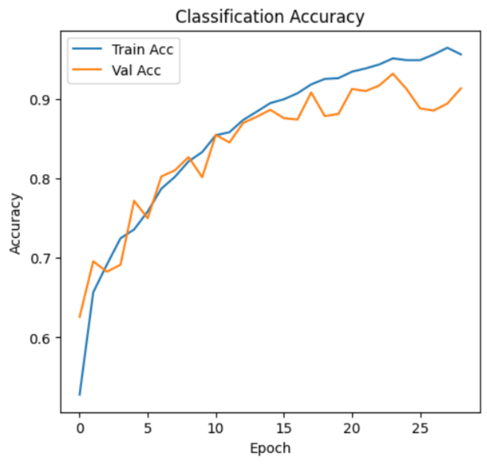


Fig. 6. Model Traning and Validation Accuracy

However, a crucial observation regarding the validation loss (val_loss) and validation accuracy (val_class_output_accuracy) dictated our training strategy. While val_loss initially improved, hitting its lowest point of 0.2105 at Epoch 24, with a solid val_class_output_accuracy of 0.9318, performance on unseen data then began to diverge. At Epoch 25, val_loss jumped to 0.3462, and val_class_output_accuracy fell to 0.9126. This concerning trend continued: by Epoch

27, val_loss reached 0.3777, and val_class_output_accuracy dropped to 0.8855. This happened even as training loss continued to decrease (e.g., 0.1291 at Epoch 27) and training accuracy remained high (e.g., 0.9576 at Epoch 27). This classic pattern—where the model improves on training data but degrades on validation data—is a definitive sign of overfitting. As a result, our early stopping mechanism activated, halting training to preserve the model's best generalization performance and prevent it from memorizing noise specific to the training set.

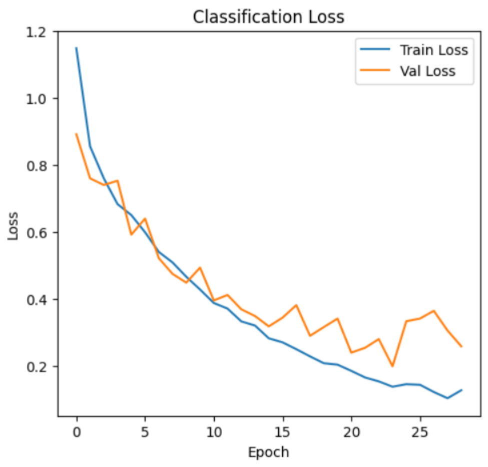


Fig. 7. Model Traning and Validation Loss

Further qualitative assessment of the model's performance was conducted through visual inspection of its predictions and an analysis of Grad-CAM heatmaps. As depicted in Figure 8, the first set of visual outputs presents the model's predicted waste class and category alongside the true labels for a selection of items, such as "GLASS", "LEAF", "METAL", and "PAPER", and their respective recyclability status. This initial visual inspection allows for a direct qualitative understanding of successful classifications and identification of specific miscategorizations, highlighting the model's ability to discern different waste types.

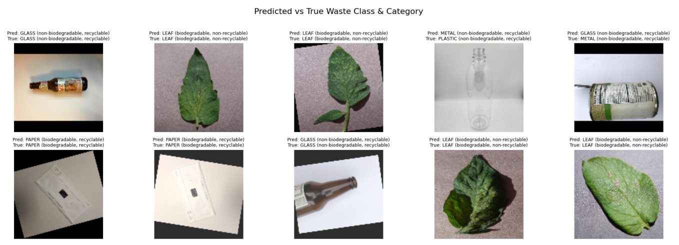

Predicted vs True Waste Class & Category
Pred: GLASS (non-biodegradable, recyclable)
True: GLASS (non-biodegradable, recyclable)
Pred: LEAF (biodegradable, non-recyclable)
True: LEAF (biodegradable, non-recyclable)
Pred: LEAF (biodegradable, non-recyclable)
True: LEAF (biodegradable, non-recyclable)
Pred: METAL (non-biodegradable, recyclable)
True: PLASTIC (non-biodegradable, recyclable)
Pred: GLASS (non-biodegradable, recyclable)
True: METAL (non-biodegradable, recyclable)
Pred: PAPER (biodegradable, recyclable)
True: PAPER (biodegradable, recyclable)
Pred: PAPER (biodegradable, recyclable)
True: PAPER (biodegradable, recyclable)
Pred: GLASS (non-biodegradable, recyclable)
True: GLASS (non-biodegradable, recyclable)
Pred: LEAF (biodegradable, non-recyclable)
True: LEAF (biodegradable, non-recyclable)
Pred: LEAF (biodegradable, non-recyclable)
True: LEAF (biodegradable, non-recyclable)

Fig. 8. Model Predictions

Complementing these individual predictions, the 10 Random Grad-CAM Heatmaps with Waste Classification (Figure 9) provided crucial interpretability. These heatmaps visually illustrate the regions of highest activation, indicating where the model focused for its classifications. While often concentrating on the waste item itself (e.g., leaves, paper), some instances, like the top-left 'glass' image, revealed background activation, offering insights into the model's feature reliance and guiding future refinements.

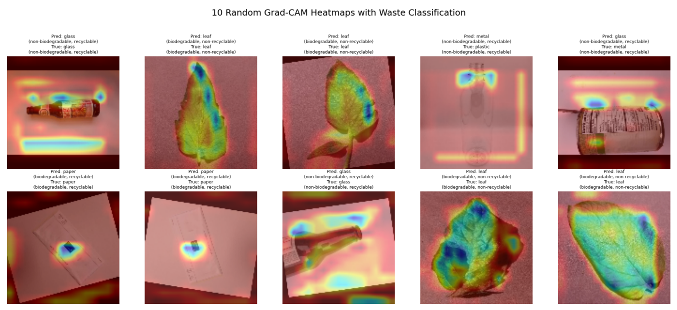

10 Random Grad-CAM Heatmaps with Waste Classification
Pred: glass (non-biodegradable, recyclable) True: glass (non-biodegradable, recyclable)
Pred: leaf (biodegradable, non-recyclable) True: leaf (biodegradable, non-recyclable)
Pred: leaf (biodegradable, non-recyclable) True: leaf (biodegradable, non-recyclable)
Pred: metal (non-biodegradable, recyclable) True: plastic (non-biodegradable, recyclable)
Pred: glass (non-biodegradable, recyclable) True: metal (non-biodegradable, recyclable)
Pred: paper (biodegradable, recyclable) True: paper (biodegradable, recyclable)
Pred: paper (biodegradable, recyclable) True: paper (biodegradable, recyclable)
Pred: glass (non-biodegradable, recyclable) True: glass (non-biodegradable, recyclable)
Pred: leaf (biodegradable, non-recyclable) True: leaf (biodegradable, non-recyclable)
Pred: leaf (biodegradable, non-recyclable) True: leaf (biodegradable, non-recyclable)

Fig. 9. Prediction Heatmap

Overall, the model performs well quantitatively, as summarized in the Classification Report (Fig 10). The model achieved an overall Accuracy of 0.9407 and a weighted F1 Score of 0.9408 across all waste categories. Examination of the per-class metrics reveals strong performance:

- The 'leaf' class demonstrated exceptional performance with a precision, recall, and f1-score of 1.00, indicating perfect classification for all 261 instances of leaves.
- 'Paper' exhibited high recall (0.97) and a precision of 0.94, yielding an f1-score of 0.96 for 243 instances of paper.
- 'Plastic' showed a high precision of 0.93 and a recall of 0.91, resulting in an f1-score of 0.92 for 244 instances of plastic.
- 'Metal' achieved strong performance with a precision of 0.96, recall of 0.90, and an f1-score of 0.93 for 261 instances of metal.
- 'Glass' also performed well, recording a precision of 0.84, recall of 0.91, and an f1-score of 0.87 for 137 instances of glass.

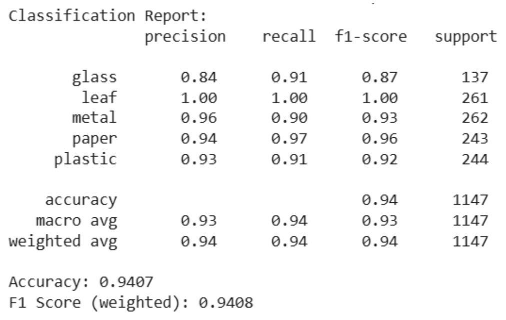
The macro avg and weighted avg f1-scores of 0.93 and 0.94 respectively further underscore the model's consistent and high performance across all 1147 instances, despite varying support sizes.

Overall, combining these quantitative and visual analyses provides a comprehensive understanding of the model's strengths, weaknesses, and decision-making processes, confirming its effectiveness and reliability for waste classification tasks.

V. Conclusion

This study developed a real-time, high-accuracy object detection model for waste classification using TensorFlow and convolutional neural networks (CNNs). Four distinct models were evaluated to determine the most effective architecture: a Baseline CNN, a ResNet-50 model with transfer learning, and two YOLO-inspired CNN models.

Among the models tested, the final YOLO-inspired architecture proved to be the most effective. Although training was scheduled for 50 epochs, it concluded early at the 29th epoch



```
Classification Report:
              precision    recall  f1-score   support

       glass       0.84      0.91      0.87       137
        leaf       1.00      1.00      1.00       261
       metal       0.96      0.90      0.93       262
       paper       0.94      0.97      0.96       243
     plastic       0.93      0.91      0.92       244

    accuracy                           0.94      1147
   macro avg       0.93      0.94      0.93      1147
weighted avg       0.94      0.94      0.94      1147

Accuracy: 0.9407
F1 Score (weighted): 0.9408
```

Fig. 10. Classification Report

due to the EarlyStopping callback, indicating efficient convergence. The model was trained using a batch size of 8 and 120×120-pixel images. Training tools such as TensorBoard, ModelCheckpoint, and EarlyStopping played a significant role in achieving high accuracy and strong generalization. The final model achieved an accuracy of 94.07% in classifying various waste types and demonstrated strong potential for enhancing residential waste segregation and operational efficiency in waste management facilities, thereby contributing to more sustainable waste processing practices.

Despite these promising results, there are notable limitations. Hardware constraints, specifically the limited GPU and RAM resources available on Google Colab, necessitated reducing image resolution to 120×120 pixels to avoid system crashes. These adjustments may have constrained the model's overall learning capacity and performance.

Additionally, future research may investigate the use of grayscale photos, especially for deployment in lightweight, resource-constrained locations like edge devices, even though color images were crucial in this study for differentiating between waste categories. Classification accuracy and model robustness may be further improved by further model architectural enhancements, such as adding more convolutional layers, fine-tuning hyperparameters more thoroughly, and carrying out more training iterations.

The dataset used in this study consisted of general waste categories available at the time of development. To ensure broader applicability, future iterations should incorporate more diverse and up-to-date real-world images, reflecting a wider range of waste types and conditions.

For garbage categorization and segregation systems, the suggested model shows a high degree of practical applicability despite these drawbacks. Its accuracy and real-time detection capabilities make it an invaluable tool for encouraging more effective domestic garbage sorting and facilitating automation in waste management processes, both of which eventually assist long-term sustainability objectives.

References

[1] G. E. Sakr, M. Mokbel, A. Darwich, M. N. Khneisser, and A. Hadi, "Comparing deep learning and support vector machines for autonomous waste sorting," in *Proceedings of the IEEE International Multidisciplinary Conference on Engineering Technology*, 2016.

[2] A. Carullo and M. Parvis, "An ultrasonic sensor for distance measurement in automotive applications," *IEEE Sensors Journal*, vol. 1, no. 2, p. 143, 2001.

[3] Mayo, Bayon, and F. M. Martín, "A comparison of several machine learning techniques for the centerline segregation prediction in continuous cast steel slabs and evaluation of its performance," *Journal of Computational and Applied Mathematics*, 2017.

[4] M. G. Flores and J. B. T. Jr, "Literature review of automated waste segregation system using machine learning: A comprehensive analysis," *International Journal of Simulation: Systems, Science and Technology*, vol. 20, no. S2, 2020. [Online]. Available: https://ijssst.info/Vol-20/No-S2/paper15.pdf

[5] S. Singh *et al.*, "Waste segregation system using artificial neural networks," *Helix The Scientific Explorer*, vol. 7, no. 5, pp. 2053–2058, 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014, [Online]. Available: https://arxiv.org/abs/1409.1556.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015, [Online]. Available: https://arxiv.org/abs/1510.00149.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788, [Online]. Available: https://arxiv.org/abs/1506.02640.

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018, [Online]. Available: https://arxiv.org/abs/1804.02767.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, [Online]. Available: https://www.deeplearningbook.org/.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] J. Brownlee, "Train, validation and test sets in machine learning," https://machinelearningmastery.com/train-validation-test-datasets/, 2018.

[14] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, [Online]. Available: https://arxiv.org/abs/1609.04747.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, [Online]. Available: https://arxiv.org/abs/1409.4842.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, [Online]. Available: https://arxiv.org/abs/1512.03385.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, [Online]. Available: https://arxiv.org/abs/1704.04861.

[20] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019, [Online]. Available: https://arxiv.org/abs/1905.11946.