An Efficient Computer Vision and Machine Learning Model for Real-Time Litter Detection on Raspberry Pi

[1]Bernard Okyere and [2]Linus Antonio Ofori Agyekum

[1,2]Dept of Electrical and Electronic Engineering, Kumasi Technical University, Ghana

Abstract

Improper waste disposal and littering pose significant environmental challenges in urban areas worldwide. Existing regulations and technologies often fall short of effectively addressing this issue. To provide a cost-effective solution, this paper demonstrates the feasibility and practicality of using computer vision and machine learning for litter detection, contributing to environmental preservation. This involves designing and deploying a Convolutional Neural Network (CNN) and Computer Vision model on a Raspberry Pi for real-time litter detection. It employs the SSD-Mobilenet-v2-FPNLite-320x320 Model for improved accuracy in detecting different types of litter. An overall mean average precision (mAP) of 76.5%, indicates the system's effectiveness in detecting and classifying litter objects. Furthermore, the deployed model achieved significant feet by identifying and classifying various types of litter objects, including plastic bottles, paper, and polythene, achieving high precision and recall scores for these classes, and facilitating prompt detection and response in practical settings. This research provides a valuable solution for tackling littering and waste management challenges in public spaces.

Keywords: *Convolutional Neural Network, Machine Learning, Object Detection, TensorFlow.*

1.0 INTRODUCTION

Mismanaged urban waste results in urban litter, such as plastic bottles left on curbs or paper wrappers blown away from trash bins by the wind (Ballatore et al., 2022). In developing countries like Ghana, the improper handling of solid waste by residents leads to littering of public spaces such as streets and waterways. This is mostly caused by a shortage of trash bins in public spaces, a lack of public awareness of the detrimental impacts of improper waste handling on public health, and insufficient enforcement of environmental legislation (Lissah et al., 2021). One way to address this environmental problem of mismanaged urban waste and litter is through the implementation of smart technologies, which can offer a more cost-effective solution.

Littering remains a persistent problem in many areas, and existing regulations and technologies are often insufficient to fully address it (Proença & Simoes, 2020). Indeed, by offering precise and effective solutions, Artificial Intelligence (AI) technology has the potential to significantly increase the efficacy of waste management systems. For instance, AI-powered sensors and cameras can be used to detect littering in real-time and alert authorities to take prompt action. Additionally, AI-powered robots can be used to collect litter and perform other tedious tasks, thereby reducing the workload on human workers (Balaji et al., 2017; Kraft et al., 2021). These technologies can help to make waste management more effective, efficient, and sustainable, while also improving the overall cleanliness and livability of urban areas in sub-Saharan

Africa. Furthermore, AI technology may be used to streamline waste collection schedules and routes, saving time and costs.

According to the Department of Economic and Social Affairs of the United Nations, the population density of urban cities is anticipated to exceed two-thirds (68%) by 2050. In this regard, governments and stakeholders need to adopt smart solutions that reduce the effects of growing communities. To address the growing issue of solid waste generation and littering, various strategies and deployments have been implemented in recent years. In essence, smart waste management solutions will help mitigate the impact of waste on the environment.

Advancements in AI have led to the development of Computer Vision (CV) and Machine Learning modules that can address waste management and plastic waste disposal. ML, a subset of AI that deals with creating algorithms and statistical models that let computers learn from data without being explicitly programmed, enables computers to learn from a set of input data used for training. This allows computers to learn from examples, handle noisy and incomplete data, and tackle non-linear problems. Once trained, computers can make predictions and generalizations quickly and accurately (Pramod et al., 2021).

Additionally, the rapid advancements in hardware, such as sensors and Graphical Processing Units (GPUs), have greatly increased the computing speed of deep learning algorithms (Wang et al., 2021). Also, the availability of modern open-source ML and Deep Learning (DL) frameworks, such as TensorFlow and PyTorch, has made it easier to build and deploy DL models on edge platforms. These frameworks have made it possible to leverage high-performance AI technology on edge devices more easily than before (Ferm, 2020).

Object detection requires a computer to learn to recognize the appearance of the object, the context in which it appears, and the variations that may occur in terms of scale, orientation, lighting, and other factors. These models can learn more complex and abstract features from images, leading to higher accuracy and more efficient object detection (Zhao et al., 2019). Additionally, some of the recent models are capable of detecting multiple objects simultaneously, which is important for real-world applications such as self-driving cars and surveillance systems (Ferm, 2020).

1.1 Single-Stage Object Detection Algorithms

Single-stage object detection algorithms, such as YOLO (You Only Look Once) and SSD (Single Shot Detector), operate on the entire image at once, directly predicting the class and location of objects in a single pass (Liu et al., 2016; Yang & Song, 2021). They use CNN to predict the class and location of each object, which can be done much faster than the two-stage approach. While they are generally faster, single-stage algorithms can be less accurate than two-stage methods, particularly for small objects or those that are closely packed together.

The selection of an algorithm is based on the application's particular requirements, such as speed or accuracy standards, as well as the size and complexity of the objects being identified. Various approaches have been developed to achieve real-time object detection based on convolutional neural networks (Kraft et al., 2021).

1.2 CNN Model Architectures for Real-Time Detection on Edge Devices

CNNs have become popular in computer vision due to their ability to extract features from images and detect patterns, edges, and textures. These advancements in CNNs have improved the performance of learning systems and greatly expanded the applications of computer vision (Bhatt et al., 2021). The architecture of CNN typically includes an input layer, multiple convolutional and pooling layers, and fully connected layers. The pooling layers are used to decrease the dimensionality of feature maps, while the convolutional layers are used to learn and extract features from the input data. The fully connected layers are used to make predictions based on the learned features (Albawi et al., 2017; Sakib et al., 2019). Real-time object detection employs a combination of feature extraction, object proposal generation, and classification to detect and track objects. This is achieved through algorithms that combine object detection and tracking techniques. Implementing real-time object recognition applications on low-cost Edge AI platforms presents a difficult challenge (Ferm, 2020). Edge devices such as raspberry pi are resource constraints in terms of computational power and memory, which makes deployment of deep neural networks in real-time applications a big challenge unlike cloud computing and virtual environments.

Tensorflow 2 model detection zoo provides a range of pre-trained object detection models that can be used as a starting point for training custom models. These models are based on various architectures such as Faster R-CNN, Single Shot Detector (SSD), and RetinaNet, among others. The best architecture for a given project relies on several variables, including the amount of the dataset, the complexity of the object to be identified, and the desired speed of inference. Each design has benefits and limitations.

To select the right model, it is important to evaluate the trade-offs between accuracy and speed of inference. For instance, faster R-CNN provides better accuracy at the cost of slower inference time, while SSD is faster but may sacrifice some accuracy (Huang et al., 2017). The choice of architecture also depends on the number of classes to be detected and the size of the dataset. For small datasets, it is recommended to use architectures with fewer parameters to avoid overfitting. However, for larger datasets, more complex architectures may be used to improve accuracy.

1.3 Object Detection Metrics

Object detection metrics are used to evaluate the performance of object detection algorithms and systems. These metrics provide quantitative measures of how well the system can detect and localize objects in an image or video. One of the commonly used metrics is the mean Average Precision (mAP). Microsoft's common objects in context (COCO) dataset (Lin et al., 2014) is often used as a standard benchmark for this metric. A higher mAP score indicates that the model is more accurate in detecting objects in images (Padilla et al., 2020). Average Precision (AP) measures the accuracy of object detection by calculating the precision-recall curve. Precision refers to the ratio of true positive (TP) detections (correctly detected objects) to the total number of positive detections (all detected objects, whether correct or incorrect). It quantifies the system's ability to avoid false positives (FP) (incorrectly identified objects or detections that do not correspond to actual objects present in the scene). On the other hand, recall measures the ratio of TP detections to the total number of ground truth objects (the actual objects present in the scene). It captures the system's ability to detect all relevant

objects, avoiding false negatives (FN) (missed detections where the system fails to identify actual objects).

Intersection over Union (IoU) threshold is another important aspect of object detection metrics. It determines the minimum overlap required between predicted and ground truth bounding boxes to consider a detection as correct. Different IoU thresholds, such as 0.5 to 0.95, are often used to evaluate the system's performance at varying levels of strictness in bounding box alignment. These metrics play a crucial role in assessing and comparing different object detection algorithms, models, and systems, enabling researchers and developers to optimize and improve their performance.

2.0 RELATED WORKS

Several published research have examined the use of object detection models for waste management and environmental monitoring. Artificial intelligence and machine learning techniques for litter detection and management have received more attention in recent years. These works have utilized various models and architectures to develop litter detection systems that can effectively identify and classify different types of litter in real time.

Bin-e is a smart waste bin that incorporates AI technology to sort and compress plastic and paper waste, manage fill levels, and process data for efficient waste management with an accuracy of more than 90%. Machine learning technique is employed for waste sorting into different bins; however, it is not intended for litter detection, but only for segregation purposes.

(Kraft et al., 2021) proposed a cost-effective approach to detect litter and trash objects in low-altitude imagery captured by an unmanned aerial vehicle (UAV). The method utilizes deep convolutional neural networks for object detection, with a custom dataset prepared to train and evaluate the model. The team tested several embedded devices to enable the deployment of deep neural networks for real-time inference on the UAV during an autonomous patrol mission.

(Huh et al., 2021) introduced an Internet of Things (IoT)-based smart bin that utilizes sensors, image processing techniques, and spectroscopes to sort waste based on pre-defined separation rules after evaluating the materials within them. Administrators can modify these smart bins based on the intended application and manage them remotely through a wireless network. Additionally, the potential for an increased range of applications for these affordable systems exists.

The authors (Bobulski & Kubanek, 2019) presented an image processing and CNN-based system to classify plastic waste into four categories: polyethene terephthalate, high-density polyethene, polypropylene and polystyrene. The system employs a red-green-blue (RGB) digital camera and software to recognize and categorize plastic waste. However, the system is not suitable for implementation in an embedded system like a Raspberry Pi due to its computing requirements.

(Samann, 2017) introduced a low-cost design for a smart waste container suitable for small-scale use. The system employs an Arduino Nano board and an ultrasonic sensor to monitor the fill level of the container and send SMS alerts via a GSM module. No machine-learning technique was employed in this study.

3.0 METHODOLOGY

The data collection process for our custom litter detection model consisted of two folds. Firstly, manual web scraping was done to collect images of trash in the context of Ghana's surroundings. Secondly, images were taken using a smartphone camera to increase the diversity of the dataset. Data cleaning was performed to ensure data quality. Images of the types presented in the list below were discarded.

- Low-quality images.
- Duplicated images.
- Images with non-valid format (Only images with .jpg extensions were used).
- Images with similar instances.
- Non-relevant images.

3.1 Data Annotation/Labelling

The collected images were labelled by creating bounding boxes around them. The LabelImg annotation tool was used for this purpose. The number of classes used for the model was nine, which were selected based on common litter types found in Ghana's environment. These classes were polythene, sachet, plastic bottle, plastic cup, paper cup, can, paper/tissue, lid, and other trash (biscuit, toffee and paper wrappers). All annotated images come with a chosen conversion format. Some of the commonly used formats are comma-separated values (CSV) and extensible markup language (XML). The chosen format for the model was XML.

Figure 1: Dataset Annotating

3.2 Data Annotation Check

In total, 1,063 images containing 2,397 instances (number of annotated objects) of the nine classes were obtained, each having its corresponding XML file. ***Figure 2*** shows a graphical overview analysis of the dataset. The platform used for this task was Roboflow.

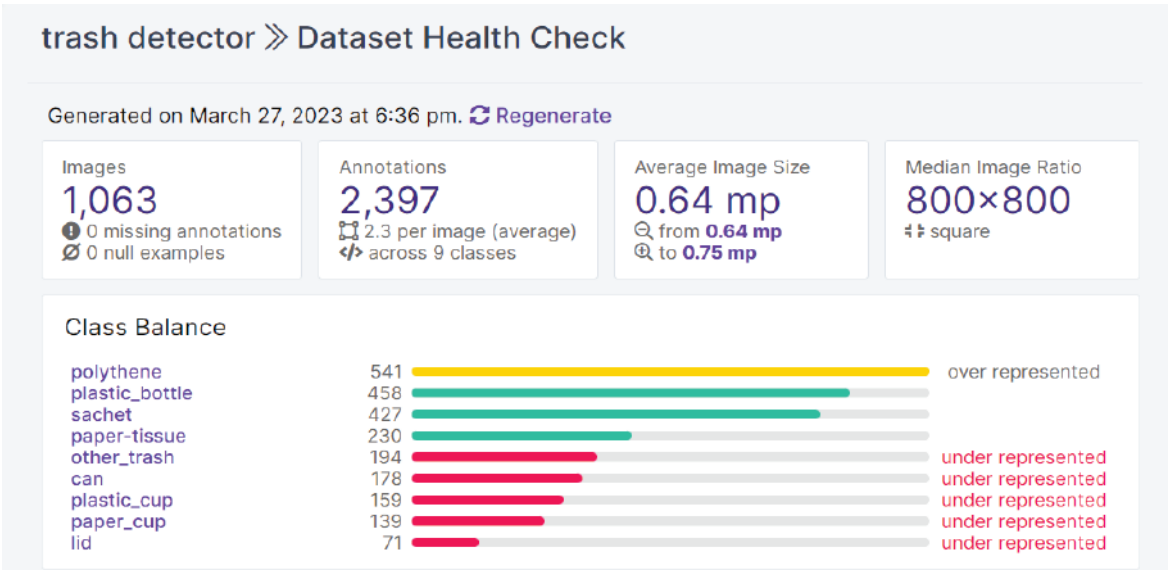

Figure 2: Dataset Annotation Overview

3.3 Model Architecture Selection

The model architecture used for training was SSD-MobileNet-V2-FPNLite-320x320. The selection was made after careful consideration of the trade-offs between accuracy and computational efficiency, which are particularly critical for real-time applications on resource-constrained devices like the Raspberry Pi. The rationale behind this choice lies in the following key factors:

- ***Computational Resources:*** The Raspberry Pi has limited computational resources, including CPU and memory. Using a smaller input size (320x320) significantly reduces the computational load, making it more feasible for real-time processing.
- ***Speed:*** SSD-MobileNet-V2-FPNLite-320x320 offers a faster inference time compared to the larger 640x640 variant. This is essential for achieving real-time performance, where the system must process frames in a timely manner.
- ***Acceptable Accuracy:*** While SSD-MobileNet-V2-FPNLite-320x320 may sacrifice a slight amount of accuracy compared to the larger model, it still provides a satisfactory level of performance for litter detection. Our preliminary experiments and validation results demonstrated that the selected model size met our requirements for the intended application.

While SSD-MobileNet-V2-FPNLite-320x320 provides a good balance between speed and accuracy, it is not without its limitations. It may not perform optimally in scenarios with highly cluttered environments or small litter objects. The model's performance can be influenced by factors such as object size, occlusions, and lighting conditions. Achieving true real-time performance may be challenging in situations with a high number of objects to detect and track. This could result in processing delays or frame drops, especially when processing video at a high frame rate. Moreover, the model's ability to detect various types of litter items may vary. It may perform better on certain types of litter (e.g., bottles, cans) compared to others (e.g., small pieces of paper or plastic).

For any object detection task, before the model is trained, the collected images need to undergo pre-processing to ensure a uniform size and quality of the images. The pre-processing

includes image resizing, normalization, and augmentation techniques to improve the model's robustness to variations in the data. The images collected for the litter detector model were resized to a resolution of 800x800 pixels. Since the SSD-Mobilenet-V2-FPNLite-320x320 pre-trained model was used as the base architecture, no other pre-processing techniques were applied to the dataset. This is because the chosen model comes with pre-processed parameters.

3.4 Model Training Process

The training of the model was performed on Google's ML research platform Google Colaboratory (Colab Notebook). Colab notebook is an interactive environment that provides access to free processing power for ML training, thus, central processing units (CPUs), graphical processing units (GPUs) and tensor processing units (TPUs) on the cloud with no configurations required.

Meticulous guidelines on how to set up the training configurations for custom model training can be found on EdjeElectronics GitHub repository. These guidelines were adopted to train our model on the colab notebook. The following are excerpts from the instructions:

1. ***Gathering and labelling of training images/dataset:*** This task was performed in *section 3.1*.
2. ***Installing TensorFlow object detection dependencies*:** Before a model can be able to train, it requires some frameworks and application programming interfaces (APIs) to run on. For our model, TensorFlow models' GitHub repository was used. This was accomplished by git-cloning it (using a command to transfer it to our colab notebook) and executing a few installation scripts.
3. ***Uploading images/dataset:*** This step involved putting all annotated images and their corresponding XML files into a single folder, compressing it to a zip file, and saving it as ***images.zip*** (performed on the local computer). The rundown below is how the content of the zip file appears. Subsequently, the file could either be uploaded directly to the colab notebook or a cloud storage (google drive) and accessed it using a command.
4. ***Preparing training data:*** In this step, we unzipped the ***images.zip*** file into a designated folder directory. Next, we split the dataset into ***Train***, ***Validation*** and ***Test*** folders. 80% of the dataset goes to training, 10% to validation and 10% to test. The images in the train folder are the ones used to train the model.

   During the training of the model, batches of images from the "train" set are passed through the neural network to predict the classes and locations of objects. The "validation" set of images is used by the algorithm to monitor the training progress and modify hyperparameters such as the learning rate. The "test" set of images is not seen by the neural network during the training process and is reserved for final human testing to assess the accuracy of the model. Subsequently, the train and validation sets were converted into 'tfrecords', a format acceptable by TensorFlow object detection API for training.
5. ***Training configuration setup:*** This step involves setting up the training parameters which include model architecture selection, number of training steps, batch size, and configuration pipeline. The model architecture selection has been discussed

already in section 3.2. The training steps are the total amount of steps to be used for training (40,000 steps were used). The batch size is the number of images to use per training step (a batch size of 16 was used). The configuration pipeline is used to store training checkpoints.

6. ***Model Training:*** Finally, the model was trained after a successful configuration setup with a model training duration of 13,500 seconds. Loss functions are crucial in training machine learning models as they measure the error between the model's predictions and the true values. Regularization loss prevents overfitting by simplifying the model, while classification loss measures the discrepancy between predicted and true class labels. Localization loss measures the difference between predicted and actual bounding box coordinates. Total loss combines regularization, classification, and localization losses to guide model training and assess convergence.

   In Figure 3, the training losses steadily decreased with each training step until reaching a plateau of around 35,000 steps. This observation indicates that the model's learning has converged, and further training may not yield a significant improvement in the losses.

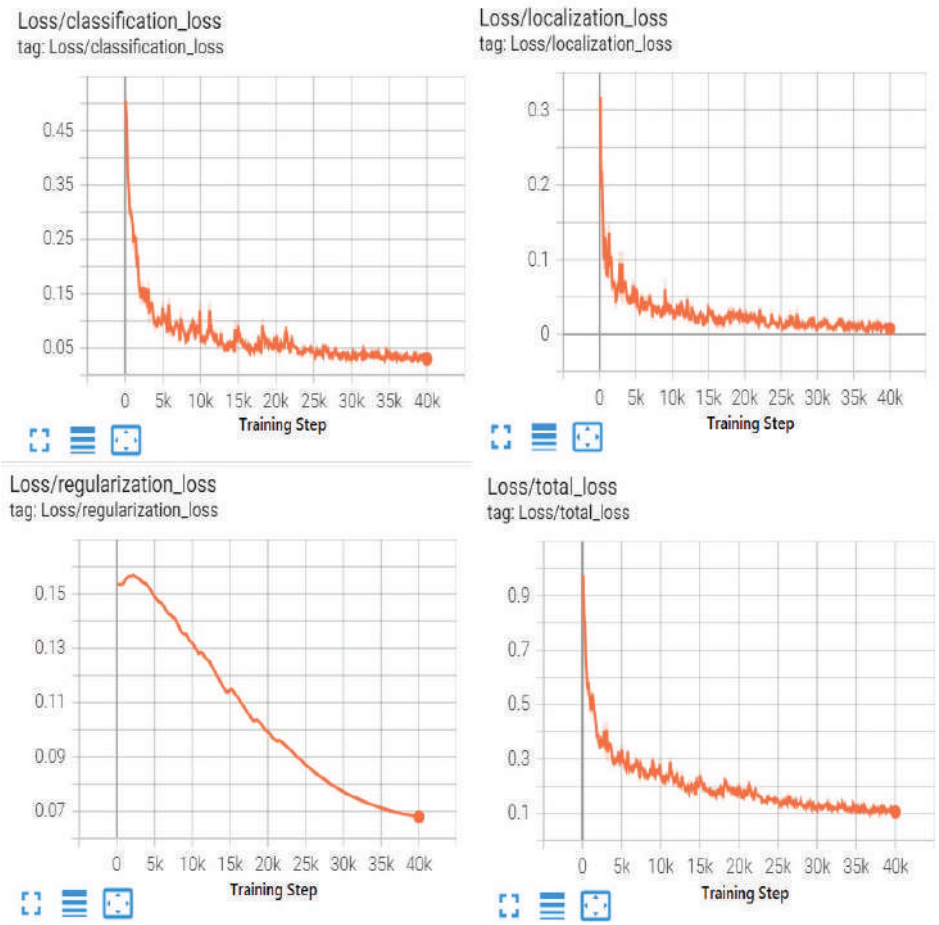

Figure 3: Training Losses

7. ***Trained model conversion:*** To be able to deploy and efficiently run the litter detector model on a raspberry pi, it needs to be exported to a lightweight version called tensorflow lite (TFLite). This is done by using acceptable commands for the conversion into the '*.tflite*' FlatBuffer format. A TFLite model requires less

processing power than regular tensorflow vision models. The TFLite model after conversion was downloaded onto a flash drive and transferred to the Raspberry Pi 4B (RP4B) via a USB port. Instructions were followed from EdjeElectronics to set up the TFLite model environment on the RP4B via the command line interface (CLI).

3.5 Trained Model Evaluation on Test Set

The test set consists of data samples that the model did not encounter during the training process. These samples are held back specifically to evaluate the performance of the trained model on unseen data and to assess its generalization ability.

Figure 4: Running Inferencing on Test Data

As depicted in ***Figures 4 and 5***, It is obvious that the litter detector model performed well in detecting litter objects within images encountered for the first time.

Figure 5: Litter Detector Model Performance Evaluation on Test Data

3.6 Performance Evaluation on Object Detection Metrics

Table 1 presents the AP values for our litter classes at various IoU thresholds. The overall mean average precision (mAP) is 76.5%. A higher mAP indicates better object detection performance. The results show varying AP values across different IoU thresholds, with generally higher APs at lower IoU thresholds. The overall mAP suggests the system's effectiveness in detecting and classifying litter objects.

Table 1: Model Performance Evaluation

Litter Class	Average Precisions at 0.5:0.95 IoU Threshold (%)										mAP (%)
	0.50	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**	
Sachet	91.2	91.2	91.2	91.2	90.7	90.7	90.7	90.7	69.3	8.7	80.6
Polythene	91.6	91.6	91.6	91.6	91.1	91.1	91.1	91.1	69.7	9.2	81.2
Can	84.3	84.3	84.3	84.3	81.9	81.9	81.9	81.9	60.8	5.1	73.1
Lid	79.1	79.1	79.1	79.1	78.2	78.2	78.2	78.2	60.3	4.9	69.4
Paper Cup	86.8	86.8	86.8	86.8	84.8	84.8	84.8	84.8	63.1	5.6	75.5
Paper/Tissue	91.3	91.3	91.3	91.3	90.9	90.9	90.9	90.9	69.5	9.0	80.7
Plastic Cup	86.5	86.5	86.5	86.5	82.9	82.9	82.9	82.9	61.7	5.6	74.5
Plastic Bottle	90.3	90.3	90.3	90.3	89.8	89.8	89.8	89.8	69.1	8.3	79.8
Other Trash	84.7	84.7	84.7	84.7	82.6	82.6	82.6	82.6	61.4	5.2	73.6
										Overall	**76.5**

3.7 Real-Time Litter Detection via Raspberry Pi Webcam

The litter detection model was successfully deployed on an RP4B, achieving an average processing speed of 1.9 frames per second (FPS). This real-time performance allows for prompt detection and response in a practical setting, making it suitable for deployment in environments such as parks, streets, or public spaces. However, an FPS as low as 1.9 means that the system processes fewer frames per second, leading to a slower response time in detecting and classifying litter objects. This can result in delays in identifying and addressing litter, which may impact the system's effectiveness in real-time applications, especially in dynamic environments with fast-moving objects or high litter accumulation rates.

Figure 6: Litter Detection via Raspberry Pi Webcam

4.0 CONCLUSION AND FUTURE WORKS

This project addresses the challenge of improper waste disposal and littering through the development of a litter detection model. By leveraging AI, computer vision, and machine learning techniques, the system achieves a high level of accuracy in real-time litter detection. The deployed convolutional neural network (CNN) and computer vision model on a Raspberry Pi 4B demonstrates its effectiveness in identifying and classifying different types of litter objects.

The system enables prompt detection and response, making it practical for deployment in various environments such as parks, streets, and public spaces. This solution provides a cost-effective and innovative approach to addressing littering and waste management challenges, contributing to environmental preservation efforts.

By successfully implementing AI and computer vision techniques, this project demonstrates the feasibility and practicality of utilizing these technologies for efficient litter detection. The findings showcase the system's capability to detect and classify litter objects, providing valuable insights for environmental conservation.

Overall, the litter detection model presented in this project offers a valuable tool for mitigating the impacts of littering and promoting cleaner and healthier urban environments. It highlights the potential of smart technologies and AI-driven solutions in creating sustainable waste management practices and emphasizes the importance of continued innovation in environmental preservation.

For future works, there are several recommendations to further enhance the litter detection capabilities:

- *Expand the dataset:* Collect and annotate a larger and more diverse dataset of litter images to improve the model's generalization and accuracy. This can include various types of litter in different environments and lighting conditions.
- *Introduce robotic functionalities:* Explore the possibility of adding autonomous navigation and manipulation capabilities to litter detection. The robot should be capable of navigating complex environments, avoiding obstacles, and physically collecting and disposing of the detected litter.
- *Implement real-time feedback:* Integrate real-time feedback mechanisms, such as audio or visual cues, to provide immediate feedback to users about the detected litter. This can enhance user engagement and encourage more proactive litter disposal.
- *Evaluate and optimize system efficiency:* Conduct further research to optimize the system's efficiency, including optimizing computational performance, power consumption, and sensor integration. This will ensure the robot's practicability and sustainability in real-world scenarios.

REFERENCES

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6.

Balaji, D. D., MEERA, S., Banu, F. A., Priya, M., ShinySherlin, C., & Sathyapriya, K. (2017). Smart trash can using internet of things. *International Journal Of Advanced Research Methodology In Engineering & Technology*, *1*(3).

Ballatore, A., Verhagen, T. J., Li, Z., & Cucurachi, S. (2022). This city is not a bin: Crowdmapping the distribution of urban litter. *Journal of Industrial Ecology*, *26*(1), 197–212.

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., & Ghayvat, H. (2021). CNN variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, *10*(20), 2470.

Bobulski, J., & Kubanek, M. (2019). Waste classification system using image processing and convolutional neural networks. *Advances in Computational Intelligence: 15th International Work-Conference on Artificial Neural Networks, IWANN 2019, Gran Canaria, Spain, June 12-14, 2019, Proceedings, Part II 15*, 350–361.

Ferm, O. (2020). *Real-time Object Detection on Raspberry Pi 4: Fine-tuning a SSD model using Tensorflow and Web Scraping.*

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & others. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7310–7311.

Huh, J.-H., Choi, J.-H., & Seo, K. (2021). Smart trash bin model design and future for smart city. *Applied Sciences*, *11*(11), 4810.

Kraft, M., Piechocki, M., Ptak, B., & Walas, K. (2021). Autonomous, onboard vision-based trash and litter detection in low altitude aerial images collected by an unmanned aerial vehicle. *Remote Sensing*, *13*(5), 965.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755.

Lissah, S. Y., Ayanore, M. A., Krugu, J. K., Aberese-Ako, M., & Ruiter, R. A. C. (2021). Managing urban solid waste in Ghana: Perspectives and experiences of municipal waste company managers and supervisors in an urban municipality. *PloS One*, *16*(3), e0248392.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37.

Padilla, R., Netto, S. L., & Da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. *2020 International Conference on Systems, Signals and*

Image Processing (IWSSIP), 237–242.

Pramod, A., Naicker, H. S., & Tyagi, A. K. (2021). Machine learning and deep learning: Open issues and future research directions for the next 10 years. *Computational Analysis and Deep Learning for Medical Care: Principles, Methods, and Applications*, 463–490.

Proença, P. F., & Simoes, P. (2020). Taco: Trash annotations in context for litter detection. *ArXiv Preprint ArXiv:2003.06975*.

Sakib, S., Ahmed, N., Kabir, A. J., & Ahmed, H. (2019). *An overview of convolutional neural network: Its architecture and applications*.

Samann, F. E. F. (2017). The design and implementation of smart trash bin. *Academic Journal of Nawroz University*, *6*(3), 141–148.

Wang, R., Wang, Z., Xu, Z., Wang, C., Li, Q., Zhang, Y., Li, H., & others. (2021). A real-time object detector for autonomous vehicles based on YOLOv4. *Computational Intelligence and Neuroscience*, *2021*.

Yang, K., & Song, Z. (2021). Deep learning-based object detection improvement for fine-grained birds. *Ieee Access*, *9*, 67901–67915.

Zhao, Z.-Q., Zheng, P., Xu, S., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232.