Received 10 September 2024, accepted 3 November 2024, date of publication 11 November 2024,
date of current version 4 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3495982

RESEARCH ARTICLE

An Integrated Learning Approach for Municipal Solid Waste Classification

HIEU M. TRAN, TUAN M. LE, HUNG V. PHAM, MINH T. VU, AND SON VU TRUONG DAO
School of Science, Engineering and Technology, RMIT University Vietnam, Ho Chi Minh City 700000, Vietnam

Corresponding author: Son Vu Truong Dao (son.daovutruong@rmit.edu.vn)

This work was supported by the RMIT Vietnam Research Grants-2024, administered by the Office for Research and Innovation.

ABSTRACT The main objective of this study is to develop and evaluate an effective integrated learning approach for the automatic classification of Municipal Solid Waste using the TrashBox dataset, comprising 17,785 images, to improve the sorting of recyclable waste materials, reduce landfill usage, and promote sustainable environmental practices. Initially, four deep learning models—DenseNet161, ResNet152, and MobileNetV3 variants—are explored to determine the most suitable feature extraction method. During the feature selection phase, three metaheuristic algorithms—Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), and Harris Hawk Optimization (HHO)—are applied to filter out irrelevant features and retain significant ones. These selected features are then fed into machine learning classifiers—Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), and K-Nearest Neighbor (KNN)—for final predictions. The DenseNet161-HHO-SVM combination outperforms other models in this study, achieving the highest accuracy and lowest execution time. This integrated approach also demonstrates superior performance (97.45%) compared to previous state-of-the-art models on the same dataset, with the data processing and method integration phases having substantial impacts.

INDEX TERMS Feature selection algorithms, integrated learning, trash system.

I. INTRODUCTION

Municipal Solid Waste (MSW) management presents a significant challenge for cities globally, impacting countries across the income spectrum, from low- to high-income nations [1]. The United States, Germany, and Japan have emerged as the top producers of municipal solid waste in recent years [2]. Although economic growth rates vary, many nations still lack adequate infrastructure to manage waste effectively [3]. This shortfall has led to a massive increase in garbage production that current systems often struggle to handle, largely due to rapid industrialization, urbanization, and population growth in metropolitan regions. For instance, Bangladesh generates approximately 25,000 tons of waste daily, with Dhaka alone contributing over 4,500 tons [4], [5]. Similarly, in Danang, Vietnam, accelerated development has raised the city's daily solid waste output to over 75 tons, with the expanding tourism industry responsible for 65% of this total [6]. Globally, an estimated 2.01 billion tons of MSW are recorded each year, a figure projected to increase to 3.40 billion tons by 2050 [7]. Insufficient waste management infrastructure often leads to disposal practices that harm both the environment and public health, result in biodiversity loss, reduced water quality, and climate change [8]. Moreover, elevated rates of cancer and birth defects have been reported in communities near incineration facilities and landfills [9].

In the broader context of resource recycling, environmental protection, and societal well-being, waste classification is essential. Solid waste is generally divided into two main categories: hazardous and non-hazardous. Hazardous waste includes domestic hazardous waste, biosolids, and railroad ties, while non-hazardous waste encompasses biogenic organic, inorganic, and mixed waste [10]. The composition of waste varies significantly by national income level: organic waste dominates in low- to middle-income countries, while high-income nations tend to produce more metal, glass, and paper waste [11]. Notably, e-waste accounted for 46.4% of the world's total in 2019, and this category has been increasing alongside the rise in electronic device usage [12], [13], signaling an environmental health risk. Moreover,

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Li.

the expansion of medical services has led to a marked increase in medical waste, with the United States generating approximately 3.5 million tons per year [14], [15], presenting an issue that warrants attention.

Effective MSW management requires implementing Integrated Waste Management (IWM) strategies that are tailored to each nation's unique waste profile. IWM combines source reduction, recycling, composting, and landfilling to mitigate waste's environmental impact [16]. However, precise waste classification is critical for these processes to function efficiently. In this study, we classify seven distinct types of solid waste—cardboard, paper, metal, glass, plastic, e-waste, and medical waste—using a combination of advanced techniques. Accurate categorization is crucial for directing waste to appropriate processing methods, thereby enhancing the overall efficiency of waste management systems [17]. Our approach proposes an optimized integrated learning model that refines both feature extraction and selection by incorporating deep learning models, metaheuristic algorithms, and data augmentation. This approach significantly improves classification accuracy by using a Support Vector Machine classifier, Harris Hawk Optimization for feature selection, and DenseNet161 for feature extraction. Our evaluations show that this method outperforms existing state-of-the-art models [18] in accuracy, owing to the complexity and refinement of our approach's structure.

II. RELATED WORKS

The application of Machine Learning (ML) and Deep Learning (DL) has expanded widely in recent years, demonstrating effectiveness and potential across numerous fields [19], [20], [21], [22], [23]. These advancements have significantly improved both accuracy and sorting rates for municipal solid waste. For example, Masand et al. [24] developed an architecture based on EfficientNet to enhance classification accuracy and reduce model complexity. They combined four datasets—TrashNet, OpenRecycle, TACO, and Waste Classification—yielding a total of 8,135 images, and achieved 91.87% accuracy overall and 98% specifically on TrashNet. Their optimization incorporated adaptive gradient clipping, which improved efficiency in high-loss areas.

Other studies have employed optimized architectures and hybrid models to increase classification precision on TrashNet. In 2018, Cenk Bircanoğlu et al. introduced RecycleNet [25], which achieved 95% testing accuracy by optimizing a deep CNN architecture for recyclable waste classification and reducing parameter count by over 57% in their 121-layer network. In 2019, Adedeji and Wang [26] applied a multi-class SVM approach using ResNet50 for feature extraction, achieving an impressive 87% accuracy on the same TrashNet dataset. Hybrid techniques have also been explored by Chu et al. [27] and Zhou et al. [28] in 2018 and 2022 respectively. Chu's research demonstrated that combining CNN-based algorithms with multilayer perceptrons could achieve 90% classification accuracy, while Zhou's work successfully classified eight classes of medical waste with 97.2% accuracy and F_1-score across five-fold cross-validation by integrating the ResNeXt model with transfer learning. These studies underscore the effectiveness of Multilayer Hybrid Systems and CNNs in enhancing waste classification accuracy.

For lesser-known datasets that have nonetheless achieved high accuracy in recent years, Vo et al. [29] introduced the ResNext-based DNN-TC model, which classified 5,904 images from the Vietnam (VN Trash) dataset into organic, inorganic, and medicinal categories. The DNN-TC model achieved 94% accuracy on TrashNet and 98% on VN Trash, surpassing other leading algorithms of that time. In 2023, Yang et al. [30] advanced the field by combining garbage classification with object recognition using ResNet and MobileNet for training and testing, and YOLOv5 for object detection. Their Consensus Voting Algorithm (CVA) enhanced classification discrimination by more than 2%, achieving 98% accuracy compared to the previous model's 95%. In 2020, a research team in China led by Ming Zeng introduced PublicGarbageNet, a dataset comprising 10 sub-classes and 10,624 images, along with a garbage classification system based on a CNN architecture [31]. Their model achieved an impressive 96.35% accuracy after extensive optimizations, including label smoothing, learning rate adjustments, data augmentation, and backbone enhancements.

Additionally, the GNet model, introduced in 2021 [32], utilized an augmented MobileNetV3 and transfer learning for waste classification. This approach was benchmarked against five models—VGG16, InceptionV3, MobileNetV3, DenseNet121, and ResNet34—using the Huawei Waste Classification Challenge Cup dataset, which categorizes waste into household, kitchen, recyclable, and hazardous categories. GNet achieved a classification accuracy of 92.62% with a processing time of 0.63 seconds. MobileNetV3 scored 86.34%, followed by ResNet34 (80.63%), InceptionV3 (77.12%), DenseNet121 (70.47%), and VGG16 (63.44%). Further progress was made in 2022 with the development of Garbage Classification Net (GCNet) by Wei Liu et al. [33]. GCNet, a unique model for trash image recognition, was built using a dataset of 41,650 images sourced online, classified into four categories: other waste, hazardous waste, kitchen waste, and recyclable waste. GCNet integrates Vision Transformer, DenseNet, EfficientNetV2, VGG Net, and ResNet for feature extraction, achieving a top accuracy of 96.75% with Vision Transformers. This was closely followed by DenseNet (96.40%), EfficientNetV2 (96.12%), VGG Net (93.77%), and ResNet (93.38%). These studies reflect significant advancements in waste classification technology, indicating promising developments in both research and application of waste classification systems for the near future.

III. METHODOLOGY

This research presents an integrated learning approach for classifying municipal solid waste using the augmented

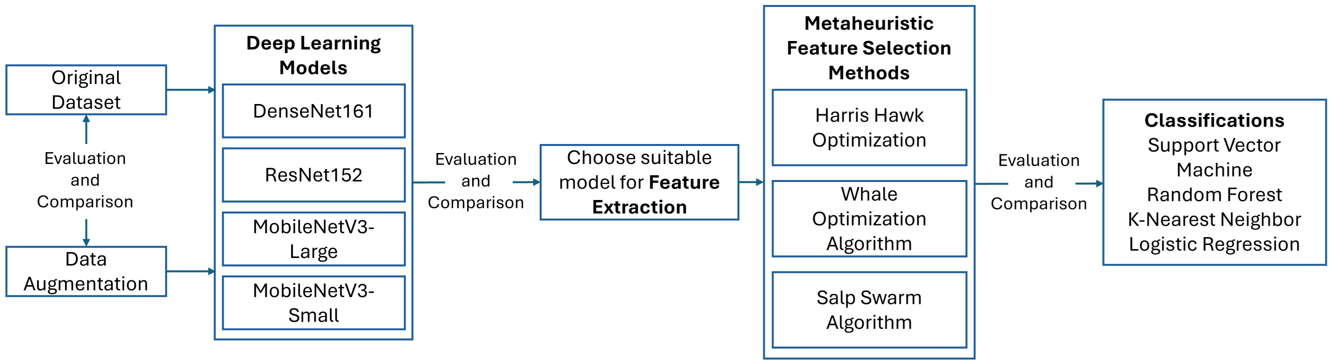

FIGURE 1. Process workflow of this research.

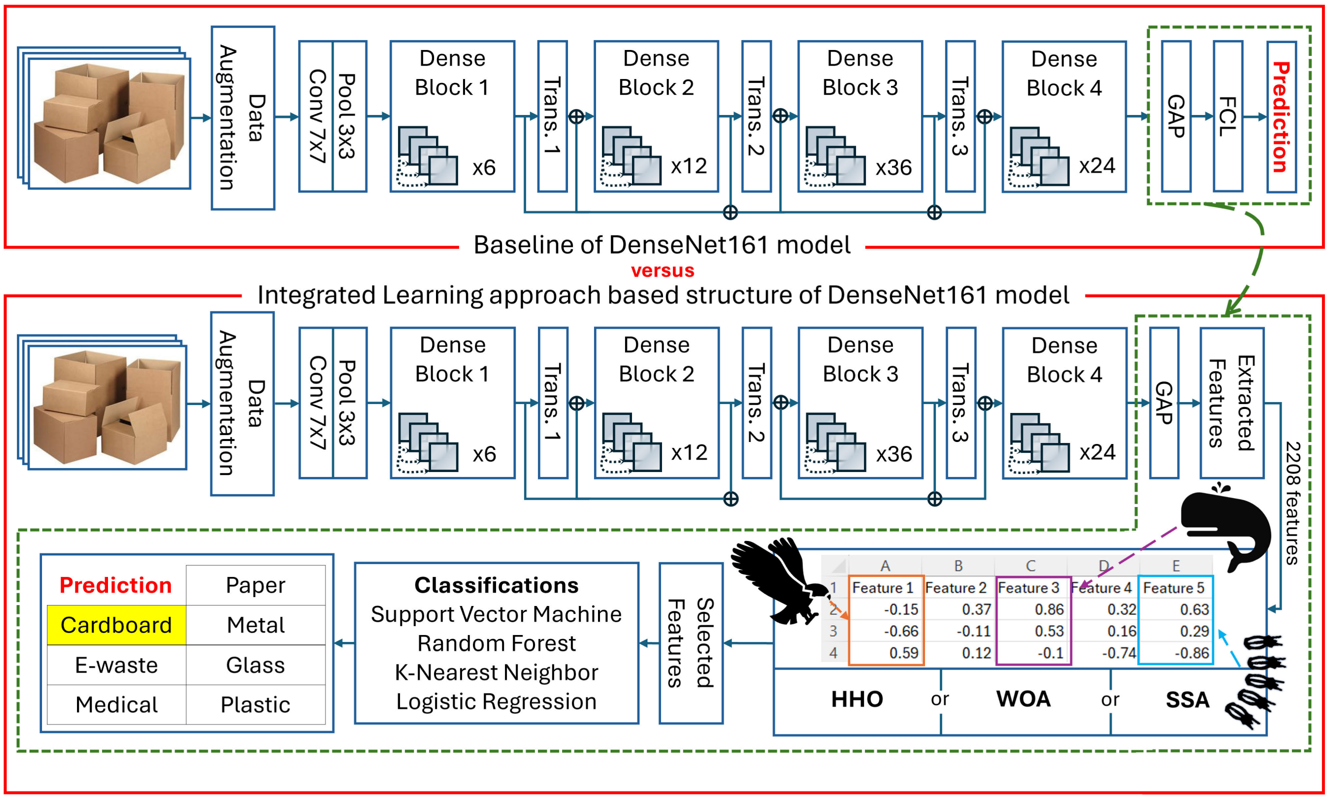

FIGURE 2. Baseline of integrated learning approach based structure of deep learning model (DenseNet161), metaheuristic feature selection and machine learning model.

version of TrashBox dataset. The overall workflow is illustrated in Figure 1, where four deep learning models—DenseNet161, ResNet152, MobileNetV3-Large, and MobileNetV3-Small—are trained using both the original and augmented datasets. The outcomes of these models are compared to demonstrate the importance of data augmentation step in the data processing pipeline and to identify the model that performs the best. The results of performance metrics (accuracy, precision, recall, F_1-score), training time, execution time and confusion matrices are considered to determine the most suitable solutions for each step in 1. As depicted in Figure 2, the DenseNet161 model, which shows the highest performance, is utilized to extract features from its Global Average Pooling layer. These extracted features are then used as inputs for the next step as shown in our proposed integrated learning approach at the bottom of Figure 2, where key features are selected by feature selection based metaheuristic algorithms from 2,208 extracted features and subsequently classified using machine learning models.

A. DATASET, AUGMENTATION AND SPLITTING

1) TrashBox DATASET OVERVIEW

In 2022, Nikhil Venkat Kumsetty et al. created a new dataset named TrashBox in order to overcome the shortcomings in the datasets that are currently available for waste identification and classification [34], consisting of 17,785 images across seven distinct categories which are briefly shown in Table 1 and Figure 3. This initiative was driven by a critical evaluation of benchmark datasets such as TrashNet (containing 2,527 images) [35] or the TACO trash dataset (with 1,500 images) [36], which highlighted significant deficiencies in both the number of images and the representation of essential waste types. They also identified that the problem relates to the lack of appropriate images to classify the trash and also the absence of important categories of trash such as e-waste and medical wastes which are integral in the waste management.

FIGURE 3. Overview of images in TrashBox dataset.

TrashBox was meticulously developed to include a diverse array of trash objects typically found in various environments, covering seven main categories: The wastes include; medical wastes, electronic wastes, glasses, plastics, carboard, papers and metals. Dataset creation process meant following a proper systematic procedure to gather data where the images of the dataset were collected and compiled from standard internet sources. These kinds of images were pre-processed into resizing, changing aspect ratios, cropping, improving qualities, noise reduction, and labeling. The proposed Trash-Box has proven as one of the biggest and highly diversified set compared to the benchmarks present in the literature making a significant improvement in the trash detection and classification area for deep learning algorithms. In 2024, Ahmed Khan, [18] has stated about their experiments using ten different deep learning methods for TrashBox dataset and the highest testing accuracy of 89.62% has been achieved from the ResNeXt-101 [37].

TABLE 1. TrashBox dataset overview.

Classes	No. of images
Cardboard	2,413
E-waste	2,937
Glass	2,528
Medical	1,957
Metal	2,586
Paper	2,695
Plastic	2,669

2) DATA AUGMENTATION AND SPLITTING

To obtain a large number number of samples for training and gain a diverse view of angles to display the images, this study also augmented the dataset. The data augmentation process involved applying both horizontal and vertical rotations to all the images (example in Figure 4), resulting in a threefold increase in the total number of images compared to the original dataset. Consequently, the dataset expanded from 17,785 images to 53,346 (after removing 3 invalid images). Additionally, the dataset was divided for training purposes in deep learning models and evaluations, with each class being randomly divided into 80 percent for training and 20 percent for testing.

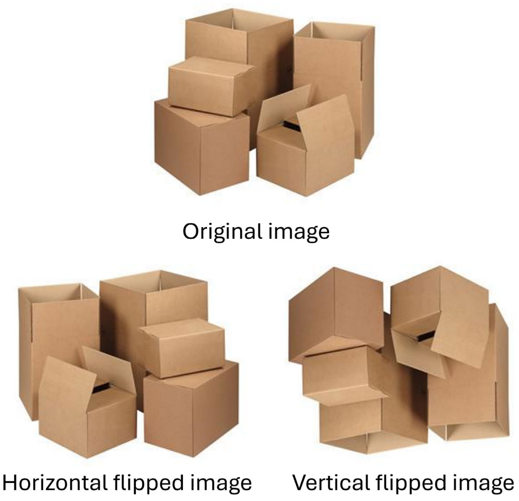

FIGURE 4. Original images and its horizontal, vertical flipped versions.

B. DEEP LEARNING MODELS-BASED CNN ARCHITECTURE

1) DenseNet161 MODEL

DenseNet's [38] architecture is designed to balance depth, efficiency, and performance which shown in the upper of Figure 2. It begins with the first set of convolutional (Conv) which takes the input image format (3, 224, 224) RGB and convolves it with a kernel size of 7×7 this while reducing the spatial dimensions of the image and increasing feature maps to 64. Next, batch normalization, ReLU activation function, and max pooling are implemented to continue the reduction of spatial size to deal with the computations deepen the network.

The modification of DenseNet is with the help of dense blocks where feature maps of all the layers are concatenated with the previous layers' output. This helps reuse of features and resolves vanishing gradient problem which is a concern in deep learning. For instance, in DenseNet161 model the first dense block produces the tensor of shape: (256, 56, 56). The number of feature maps rises as the network advances, and spatial dimensions are reduced through transition layers, and these include the 1×1 convolutional layer that forms a bottleneck and the 2 × 2 average pooling layer. In the final stages, DenseNet transitions to a compact representation suitable for classification. For instance, after the last dense block in DenseNet161, a global average pooling (GAP) layer reduces the feature map to a vector of size (2208,), which is then passed through a fully connected layer and softmax activation to produce a probability distribution over the classes.

DenseNet has several versions: DenseNet121, DenseNet169, DenseNet201, and DenseNet161, each differing in the number of layers and growth rates. DenseNet121 has 121 layers with a growth rate of 32, producing 1024 feature maps at the final stage. DenseNet169 has 169 layers and ends with 1664 feature maps. DenseNet201 has 201 layers and generate 1920 feature maps. DenseNet161 which has 161 layers and a higher growth rate of 48, leading to the largest feature map of 2208. However, DenseNet161 is adopted in this research due to the higher growth rate, as well as the richer feature representations it owns. That is why it is especially suitable for pattern recognition in images and offers a very detailed analysis of them. However, DenseNet161 has fewer layers than DenseNet201, but the structure of this network is tuned for performance, which makes this network more preferable in case of sufficient computational power.

2) ResNet152 MODEL

ResNet152 [39] is a deep convolutional neural network specifically developed to address the degradation problems that emerge with increasing network depth. The architecture begins with a 7 × 7 convolutional layer, which is then accompanied by batch normalization, ReLU and pooling layer. These operations are used for dimensions shrink and control inputs to make them ready for further feature extraction in the next layers.

The main idea in ResNet is separated with the use of residual blocks wherein the network directs shortcut connections through some of the convolutional layers, making the network to learn residual functions and not direct mappings. This architecture greatly improves the backpropagation of gradients in order to facilitate the training of very deep networks. ResNet152 is made up of 4 residual blocks (as shown in Figure 5) which include respectively 3, 8, 36 and 3 residual groups for block number 1, 2, 3 and 4. Each residual group consists of 1 Conv 1 × 1 layer for reducing spatial dimensions and increasing number of channels, one Conv 3 × 3 layer for performing major convolution and the last one Conv 1 × 1 layer for restoring spatial dimension. These 150 layers are combined with the initial layers-conv1 layer and maxpooling layer to produce 152 convolutional layers before being fed into the GAP layer. The network is concluded by a feature extraction layer that outputs class probabilities through softmax activation.

Different variants of ResNet are ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152 that also have certain depth, and numerous residual blocks. Out of these, one can particularly highlight ResNet152 as the deepest model which is capable of capturing terrifically rich and detailed features that make this model suitable where high accuracy is required for the results such as image classification. However, due to its depth and complexity, ResNet152 is designed to keep the train time some what reasonable, that also make ResNet152 best suitable for applications where accuracy is paramount.

3) MobileNetV3 MODEL

MobileNetV3 [40] is a CNN architecture that has been developed in large and small variations with the intention to work on efficiency and work on mobile, as well as embedded systems. Considering the achievements of the previous architectures like DenseNet as well as ResNet that were aimed at increasing the depth and reusing the features, MobileNetV3 takes a lighter, resource-efficient approach to work efficiently on the devices with limited resources. MobileNetV3 is defined starting from several convolutional layers stacked using a new and improved inverted residual block, though the latter originates from MobileNetV2. These blocks use depthwise separable convolutions that parse a standard convolution into two simpler operations thereby cutting computational demand. The architecture also includes what is called squeeze-and-excitation (SE) modules that help re-scale channel-wise feature responses dynamically, thus boosting the networks' ability to effectively attend to the informative features.

In general, when computing the performance of the MobileNetV3's large version, it is important to consider that the large variant of MobileNetV3 incorporates a greater number of layers and significantly more parameters in comparison to the small version. This increased complexity allows for enhanced representational capacity, enabling the model to capture more intricate features. However, it also results in higher computational demands, which may impact efficiency in resource-constrained environments. In detail, MobileNetV3 variants start with a typical Conv 3 × 3 layer, then some bottleneck blocks that are characterized by differing expansion rates and kernel sizes. Inside a bottleneck block includes 5 components which are one 1 × 1 pointwise conv layer for increasing the number of channels from 32 to 128, one 3 × 3 depthwise conv, one optional SE modules for re-calirating feature maps, one final 1 × 1 pointwise conv to reduce the number of channels back to 32 and one skip connection layer. As pointed out in Figure 6, while the SE modules are applied to all of 5 × 5 bottleneck blocks and 2 last 3 × 3 bottleneck blocks of the in structure of MobileNetV3-Large, the MobileNetV3-Small's structure just use SE modules for 5 × 5 bottleneck blocks and the first 3 × 3 bottleneck blocks. Furthermore, these variants of MobilNetV3 use 2 types of nonlinearity where HS denotes h-swish and RE denotes ReLU which also shown in the transition arrows in Figure 6. The last few layers consist of a fully connected layer that delivers the class probability using softmax activation and a global average pooling (GAP) layer. The small variant of MobileNetV3,

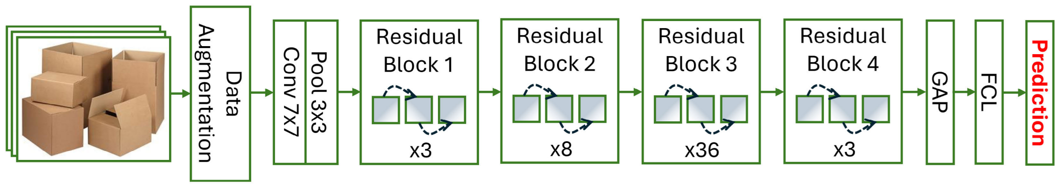

FIGURE 5. **Baseline of ResNet152 model.**

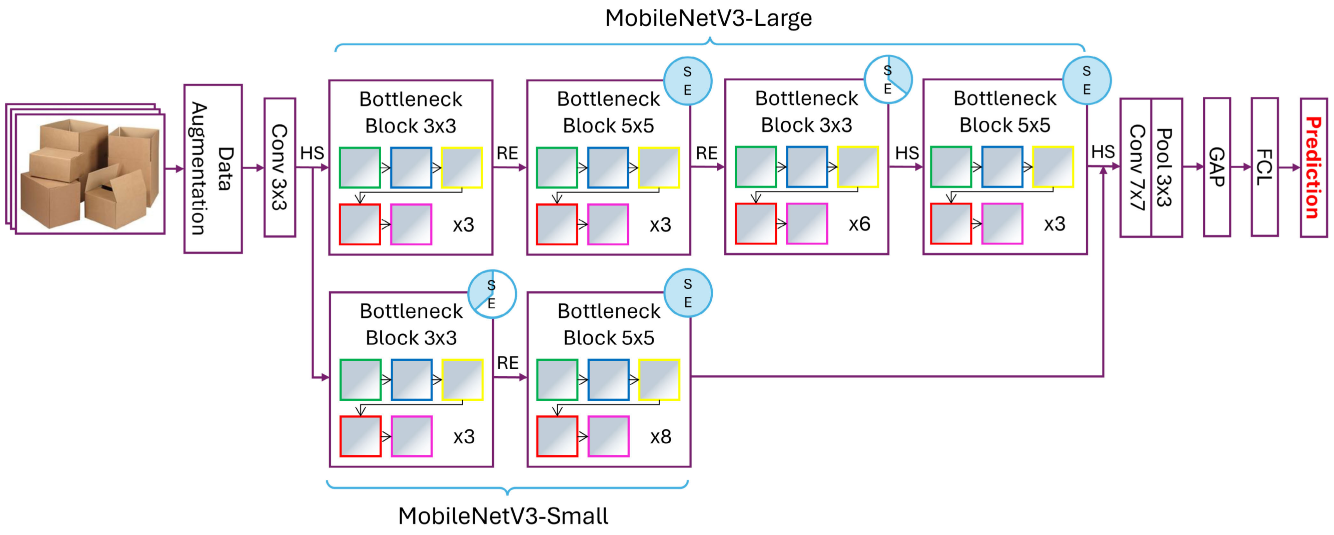

FIGURE 6. **Baseline of MobieNetV3-large and MobileNetV3-small model.**

in contrast, is further optimized to achieve even higher efficiency under significantly more constrained environment, such as those found in low-power devices. It employs fewer layers and parameters than the large version, making it more suitable for applications where computational efficiency is paramount. Despite its smaller size, it retains the core features of the large variant, including the use of inverted residual blocks and SE modules, ensuring that it still performs well in tasks where accuracy is important.

There are DenseNet and ResNet that focuses on depth and feature reuse for high accuracy and MobileNetV3 is efficient even in cases where the computational capacities are low. While DenseNet is designed with dense connections which could be beneficial for detailed feature learning, and ResNet is built with residual blocks which are also favorable for feature learning, but they are heavy in terms of computational load. MobileNetV3 comes as the compact version of the two, with moderate accuracy and much less computational requirements to serve as the perfect candidate for the mobile devices usage.

C. FEATURE SELECTION ALGORITHMS

Feature selection algorithms are crucial components in machine learning [41], [42]. They play an important role in identifying optimal feature subsets within a dataset, which helps to optimize predictive models, reduce computational costs, and simplify the interpretability of results. These algorithms aim to isolate only the most informative features, addressing issues like overfitting, reducing data dimensionality, and speeding up training. The primary goal is to maintain or improve model performance by using a more streamlined, effective set of features [43].

Feature selection algorithms are generally categorized into three types: filter methods, wrapper methods, and embedded methods. In filter methods, features are evaluated independently of any specific model using statistical measures like correlation, mutual information, or chi-square tests. These methods are computationally efficient but may lack the ability to capture feature interactions. Wrapper methods, on the other hand, use a predictive model to evaluate the performance of various feature subsets, searching for the optimal subset by training and testing the model with different features. Though more computationally demanding than filter methods, wrapper methods often yield higher accuracy. Embedded methods, such as LASSO (Least Absolute Shrinkage and Selection Operator) and decision tree-based techniques, incorporate feature selection within the model-building process itself, automatically identifying important features during training.

Metaheuristic algorithms, including Whale Optimization Algorithm [44], Salp Swarm Algorithm [45], and Harris Hawks Optimization [46], are typically classified as wrapper methods. These algorithms search through high-dimensional spaces to determine which features should be included to enhance model quality, making them highly suitable for feature selection tasks. Metaheuristic algorithms, inspired by natural processes, can efficiently explore large and diverse feature spaces to find optimal solutions within practical timeframes.

1) WHALE OPTIMIZATION ALGORITHM

The Whale Optimization Algorithm (WOA) [44] is a metaheuristic optimization algorithm developed by Mirjalili and Lewis in 2016. WOA is inspired by the unique hunting strategy of humpback whales, which use spiral bubble nets to capture prey. While foraging, humpback whales are social hunters, working together to create spirals of bubbles around schools of krill or small fish near the water surface before attacking in unison. WOA models these behaviors mathematically to address global optimization problems, making it suitable for a wide range of applications.

a: CORE PRINCIPLES OF WOA

In the WOA algorithm, a group of agents (representing humpback whales) engage in a search, circling, and hunting process to approach the optimal solution. The algorithm is structured around three main strategies: circling prey, spiral bubble-netting, and hunting (also known as the exploitation phase), and finally, exploration or prospecting for food. Each of these strategies contributes to guiding the agents (candidate solutions) toward the global optimum.

b: ENCIRCLING PREY STRATEGY

The first phase of WOA is the encircling prey strategy, which models how humpback whales identify and surround their prey. In optimization terms, the current best solution is nearest to the prey or the actual prey. Other whales then modify there position with respect to this best solution and continue doing so until all become aligned with this solution.

Mathematically, the position of a whale $\vec{X}(t)$ at iteration t is updated based on its distance to the best solution $\vec{X}^*(t)$ found so far:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)\right| \tag{1}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D} \tag{2}$$

Here, $\vec{D}$ represents the distance between the whale and the prey, indicating how far the whale is from the best solution. The vectors $\vec{A}$ and $\vec{C}$ are coefficient vectors that control the whales' movement:

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{4}$$

where $\vec{r}$ is a random vector with values in the range $[0, 1]$, and $\vec{a}$ decreases linearly from 2 to 0 over the course of iterations:

$$\vec{a} = 2 - \frac{2t}{\text{MaxIter}} \tag{5}$$

The current iteration, t, and the total number of iterations, MaxIter, are represented in this equation. The parameter a is essential in regulating the algorithm's convergence behavior as it reduces the search space as the whales get closer to the best solution.

c: BUBBLE-NET ATTACKING STRATEGY (EXPLOITATION PHASE)

In addition to describing how the humpback whale surrounds prey, the humpback whales utilize bubble nets that have the shape of a spiral to trap their prey. In WOA, this behavior is modeled through two primary mechanisms: shrinking encircling and spiral position updating. Such mechanisms enable the algorithm to properly work on the search space and improve it with the best solution.

Shrinking Encircling Method involves reducing the convergence coefficient $\vec{a}$ over time, effectively shrinking the encircling area around the prey. This restricts the searching area and gives the whales the accurate ability to locate the prey. The random coefficient $\vec{A}$ which determines the degree of the encirclement of the random point fluctuates within the range $[-a, a]$. to tighten the searching space around the global optimum. Spiral Updating Position Method mimics the spiralling movements of the whales as they approach the prey. The spiral path is mathematically represented as:

$$\vec{X}(t+1) = \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{6}$$

where $\vec{D}' = \left|\vec{X}^*(t) - \vec{X}(t)\right|$ and denotes the distance between the ith whale and the prey (best solution achieved so far), b is a constant that specifies the form of the logarithmic spiral, and l is a random value from the range $[-1, 1]$. On one hand, the speed and the direction of the whale's movement are adjusted on its spiral path while closing in its target during a coordinated attack. While optimizing the system, the algorithm utilizes either the shrinking encircling method for $p < 0.5$. During optimization stochasticity is applied and a decision of whether to use the shrinking encircling method ($p > 0.5$). This probabilistic method helps WOA to select and alternate between exploration and exploitation thus achieving more efficiency.

d: SEARCHING FOR PREY STRATEGY (EXPLORATION PHASE)

The last strategy in WOA is the search for prey phase or more commonly referred to as the exploration phase which helps in avoiding local optima and maintaining diversity of the population. During this phase, whales search for better solutions more globally to find the better solution. The exploration is guided by the coefficient A. When $|A| \geq 1$, the algorithm shifts into exploration mode, where each whale updates its position by moving away from a randomly selected whale in the population:

$$\vec{X}(t+1) = \vec{X}_{\text{rand}}(t) - \vec{A} \cdot \vec{D} \tag{7}$$

where, $\vec{X}_{\text{rand}}(t)$ is the position of a randomly whale, and $\vec{D} = \left|\vec{C} \cdot \vec{X}_{\text{rand}} - \vec{X}\right|$ is the distance between the whale and the prey. This strategy increases diversity of population and prevents the algorithm from getting trapped in local minima.

e: IMPLEMENTATION AND WORKFLOW OF WOA

WOA starts with generation of a population of whales at initial random positions in a given search space. A position of each whale is equivalent to a candidate solution and the solutions are then assessed by a predetermined objective function. The best seen solution is recorded and revised in the process of iteration production.

At each iteration, the algorithm updates the position of each whale based on the current strategy, chosen according to the value of the probability parameter p. If $p \geq 0.5$, the algorithm uses the spiral updating position method, if $p < 0.5$, the algorithm decides between the encircling prey strategy (if $|\vec{A}| < 1$) and the searching for prey strategy (if $|\vec{A}| \geq 1$).

For each updated position, the algorithm compares the newly obtained solution with the current best solution and updates it if a better solution is found, it is used to update the best solution. These iterations proceeds until a maximum number of iterations is used up or until some other termination condition has been fulfilled. All the same, several studies have indicated that WOA is capable of solving many optimization problems as presented below. Due to its clear design and easily comprehensible processes it is frequently used by researchers and practitioners. But, to a great extent, it depends on the performance of the identified exploration/exploitation trade-off, which is regulated by the dynamic adaptation of the three key strategies.

2) SALP SWARM ALGORITHM

In 2017, Mirjalili et al proposed a population based optimization algorithm called the Salp Swarm Algorithm [45]. This method is based on the phenomena of similar movements of salps – marine organisms that swim in a linked linear chain-like manner. Another discovery by SSA is the probability of the simultaneous coordinated motion of all the salps and a manner than enables these animals to effectively organize themselves in order to execute a certain task or perform a particular function in their environment appropriately.

a: CORE CONCEPTS OF SSA

SSA simulates a group of salps (agents) navigating a problem space to find optimal solutions. The algorithm operates in two primary phases: Mainly the leader salp movement and the follower salp movement. The leader salp shifts it to attractive zones and the follower salp regulates its position with respect to the leader salps and the neighboring salp to cover the leader.

b: LEADER SALP MOVEMENT

At the beginning of each iteration, the salp at the front of the chain is designated as the leader. The leader's position is updated relative to the best solution found so far in the search space, denoted as F_j for the j-th dimension. This ensures that the swarm is directed towards areas with higher potential for finding the optimal solution.

The position of the leader salp x_j^1 in the j-th dimension is updated using the following equations:

$$x_j^1 = \begin{cases} F_j + c_1 \cdot \big((ub_j - lb_j)c_2 + lb_j\big) & \text{if } c_3 \geq 0, \\ F_j - c_1 \cdot \big((ub_j - lb_j)c_2 + lb_j\big) & \text{if } c_3 < 0, \end{cases} \tag{8}$$

where:

- ub_j and lb_j are the upper and lower bounds of the j-th dimension, respectively.
- c_1 is a coefficient that decreases over time to balance exploration and exploitation.
- c_2 and c_3 are random values uniformly distributed between 0 and 1.
- F_j is the position of the best solution found so far in the j-th dimension.

The coefficient c_1 is calculated as:

$$c_1 = 2 \cdot e^{-\left(\frac{4l}{L}\right)^2}, \tag{9}$$

where L is the maximum number of iterations and l is the current iteration. As the search goes on, this coefficient aids in the algorithm's shift from early exploration to later exploitation.

c: FOLLOWER SALP MOVEMENT

The last group of salps who are referred to as followers' reorient themselves according to the salp who is right in front of them. By such reorientation, the swarm holds its shape and aligns itself with the leader in the attempt to find the optimal solution.

The position of a follower salp $\vec{X}_i(t)$ at iteration t is updated using:

$$x_j^i = \frac{x_j^i + x_j^{i-1}}{2}, \quad i = 2, 3, \ldots, N, \tag{10}$$

where N is the total number of salps in the swarm, $i \geq 2$ and x_j^i shows the position of ith follower salp in j th dimension. This equation ensures that each follower salp moves toward the midpoint between its current position and that of the salp directly ahead, promoting convergence towards the optimal solution.

d: BALANCING EXPLORATION AND EXPLOITATION

SSA provides a strategy to optimize both search and convergence by adapting the coefficient c_1. At first, c_1 takes a large value such that it permits the leader salp to delve comprehensively into the search territory. During the course of the algorithm, c_1 tends to get smaller thereby making a concentrated search on solutions that are deemed the best so far.

To complement this, andom tolerance limits defined by coefficients c_2 and c_3 alter the leader salp's movement direction is altered, which boosts the search for unvisited zones of the solution and refrain from local optima.

e: IMPLEMENTATION AND WORKFLOW OF SSA

Salps are first randomly positioned across the search space to form the population for SSA. The quantity of potential solutions for the optimization issue is equivalent to the number of salps in the search space. An objective function is used to evaluate these solutions, and the best one is provided. At each of the algorithmic steps, the salp leader changes its position and gets to move as per the movement plan that has been put in place. Notifications of movement plan of the preceding salps are then stored to follow it and update the positions of following salps. The act of analysing and elaborating the answers continues until the number of cycles set for the process is fulfilled. SSA is known for its simplicity and efficiency in solving various optimization problems. The algorithm's performance depends largely on its ability to balance exploration and exploitation, which is managed through the leader and follower salp movements.

3) HARRIS HAWKS OPTIMIZATION

Heidari et al. presented the population-based optimization algorithm known as the Harris Hawks Optimization [46] algorithm in 2019. The cooperative hunting style of Harris' hawks, a species known for its distinctive prey-hunting strategy, is the model for this algorithm. These hawks use various tactics, including sudden pounces, surprise attacks, and coordinated team efforts, to catch their prey. HHO mimics these strategies to solve complex optimization problems effectively.

a: CORE CONCEPTS OF HHO

HHO models the hunting behavior of a group of Harris' hawks (agents) as they search for the optimal solution in the problem space. The algorithm simulates different phases of hawk hunting, including exploration, surprise pounce (transition from exploration to exploitation), and diverse attacking strategies to exploit the most promising areas.

b: EXPLORATION PHASE

During the exploration phase, hawks scour the search space in an attempt to find prey—the best option. The hawks' positions are updated based on the prey's position and random variables that encourage diverse exploration.

The position of a hawk $X(t+1)$ at iteration t is updated as follows:

$$X(t+1) = \begin{cases} X_{\text{rand}}(t) - r_1 |X_{\text{rand}}(t) - 2r_2 X(t)|, & q \geq 0.5 \\ (X_{\text{prey}}(t) - X_{\text{m}}(t)) - r_3(LB + r_4(UB - LB)), & q < 0.5 \end{cases} \tag{11}$$

where:

- $X_{\text{rand}}(t)$ is the position of a randomly chosen hawk.
- $X_{\text{prey}}(t)$ is the position of the prey (best solution found so far).
- X_m is the average position of the current population of hawks
- r_1, r_2, r_3, r_4 and q are uniformly distributed random values ranging from 0 to 1 that are updated with each iteration; LB and UB represent the upper and lower boundaries of variables.

This phase allows the hawks to explore the search space broadly, increasing the likelihood of finding the global optimum.

c: TRANSITION TO EXPLOITATION: SURPRISE POUNCE

As the hawks approach the prey, the algorithm shifts from exploration to exploitation. This shift is characterized by a "surprise pounce," in which the hawks quickly alter their postures to catch their prey.

The probability of this transition is controlled by the energy of the prey E, which decreases over time according to:

$$E = 2E_0(1 - \frac{t}{T}), \tag{12}$$

where E_0 is the initial energy, t is the current iteration, and T is the maximum number of iterations. As E decreases, the hawks focus more on exploitation, honing in on the best solutions.

d: EXPLOITATION PHASE: ATTACKING STRATEGIES

During the exploitation phase, Harris Hawk Optimization employs several strategies to capture the prey, based on the prey's energy level E and the relative distance r between the hawks and the prey.

1) Soft Besiege: When $r \geq 0.5$ and $|E| \geq 0.5$, the prey still retains enough energy to make random jumps in an attempt to escape, though it eventually fails. The hawks slowly close in, causing the prey to tire out before launching a final attack. This scenario is described by the following equations:

$$X(t+1) = \Delta X(t) - E|JX_{\text{prey}}(t) - X(t)|, \tag{13}$$

$$\Delta X(t) = X_{\text{prey}}(t) - X(t), \tag{14}$$

Here, $\Delta X(t)$ represents the positional difference between the prey and the hawk at iteration t. The parameter $J = 2(1 - r_5)$, with r_5 being a random variable in the interval (0, 1), depicts the jump strength of the prey, which varies with each iteration to simulate natural movements.

2) Hard Besiege: When $r \geq 0.5$ and $|E| < 0.5$, the prey is significantly weakened, and the hawks encircle it more aggressively before striking. The position update rule is given by:

$$X(t+1) = X_{\text{prey}}(t) - E|\Delta X(t)|, \tag{15}$$

3) Soft Besiege with Progressive Rapid Dives: When $|E| \geq 0.5$ and $r < 0.5$, the prey has sufficient energy to

attempt escape, but is still softly encircled by the hawks before the final attack. This strategy is enhanced using Lévy flight (LF) to simulate the prey's erratic escape patterns and the hawks' sudden dives. The next move of the hawks is determined by:

$$Y = X_{\text{rabbit}}(t) - E|JX_{\text{rabbit}}(t) - X(t)|, \tag{16}$$

If the hawks detect more deceptive movements from the prey, they perform accelerated dives using a Lévy flight-based maneuver:

$$Z = Y + S \times \text{LF}(D), \tag{17}$$

where D is the problem's dimensionality, S is a random vector of size $1 \times D$, and LF is computed as:

$$\text{LF}(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \tag{18}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}, \tag{19}$$

where u and v are random numbers in the interval (0, 1), and β is typically set to 1.5. The hawks' final position is determined by:

$$X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)), \\ Z & \text{if } F(Z) < F(X(t)), \end{cases} \tag{20}$$

4) Hard Besiege with Progressive Rapid Dives: When $|E| < 0.5$ and $r < 0.5$, the prey is too exhausted to escape, and the hawks perform an aggressive encirclement with accelerated dives. The hawks close in on the prey's position using:

$$Y = X_{\text{prey}}(t) - E|JX_{\text{prey}}(t) - X_m(t)|, \tag{21}$$
$$Z = Y + S \times \text{LF}(D), \tag{22}$$

where $X_m(t)$ represents the mean position of all hawks at iteration t. The next position of the hawks is selected based on the better of Y or Z.

These strategies effectively update the hawks' positions, ensuring a robust convergence towards the optimal solution, thereby improving the performance and reliability of the HHO algorithm in various optimization problems.

e: *IMPLEMENTATION AND WORKFLOW OF HHO*

HHO begins with the generation of an initial group of randomly placed 'hawks' within the search space. The location of each hawk is viewed as a possible solution to the problem at hand. The algorithm evaluates these solutions using a predefined objective function and tracks the best solution found so far. During each iteration, the hawks adjust their positions according to the current phase—exploration, transition, or exploitation—based on energy levels and distance to the prey. The algorithm stops refining the present structure only when maximum number of iterations is reached or some other stopping criteria is fulfilled. HHO is a technique which has good applicability range in solving various types of optimization problems. The ability to switch between hunting strategies on the fly is very beneficial for any problem in global optimization.

4) EFFECTIVENESS IN FEATURE SELECTION ALGORITHMS

Out of all the selection algorithms examined in this study, only HHO is convenient for the objective of feature selection necessary in the high-dimensional (2208 features from DenseNet161), intricate trash classification task. Through its switch of exploration and exploitation phases it follows the hunting pattern of the hawks, while it offers the ability to escape local minimum and has a high convergence rate at the same time. This balance is paramount especially while attaining a significant decrease of a feature space as well as HHO can enhance actual subsets that in turn SVM improves classification efficiency.

Although WOA and SSA have their relative superiority; for instance, WOA can diversely explore the feature subset space, while SSA is efficient for local fine-tuning, both algorithms in general, are more suitable for simple or lower-order feature selection tasks. Among the deficiencies, the flexible exploration of WOA results in a small convergence rate especially for high-dimensional data. For its part, SSA remains feasible in overcoming high redundancy of features and can stick in local optima. On the other hand, the performance of HHO is stable regardless of the various feature dimensions, and thus eliminates dopiness and improves the performance of more comprehensive datasets such as TrashBox.

D. MACHINE LEARNING MODELS

Machine learning models are mathematical models that analyze patterns in data in order to include decisions and predictors. In contrast with conventional programming, where one's instructions are directly provided to the algorithm to execute a function, ML models independently acquire knowledge from the data to enhance the results when working on particular tasks. One of the key steps in development of the ML models is feature engineering, which consists in selecting appropriate inputs that are the features in a certain model. Simplified the sentence to clarify the role of feature selection in enhancing model performance, avoiding overfitting, and reducing training time. This must mean that learners within the realm of machine learning are well positioned to attend to features in a number of ways; where specifying the model guise can be made dependent with the kind of issue or data set being presented. Among these, popular algorithms such as Support Vector Machines, Random Forest, Logistic Regression, K-Neighbors Algorithm are used for problems of discrimination, approximation of functions or target variables, and divisions. These models aim at solving problems of classification, regression and clustering and are applied in various fields such as data science, artificial intelligence and predictive analytics.

1) SUPPORT VECTOR MACHINE

Support Vector Machine is one of the powerful supervised learning algorithms which is actually employed for the classification data sets nevertheless it plays a significant role in the field of regression as well. In brief, the purpose of SVM is to find the optimal hyperplane that would correctly classify the data points belonging to different classes present in the feature space. The dual form lies in the fact that unlike in linear programming, the objectives do not involve the values of the classes, only their differences, the width of the hyperplane between the classes being as large as possible thus generalizing well to new data.

When data is not separable using a linear technique, SVMs use kernel trick to transform data into a higher dimensions where data can be separable. Some of the kernels to be discussed include linear, polynomial, as well as the radial basis function(RBF) kernels. The decision as to the type of kernel is a critical one because it influences the model's performance quite significantly. The decision boundary in SVM is expressed as:

$$f(x) = \text{sign}(w^T x + b) \tag{23}$$

where w is the weight vector, b is the bias term, and x represents the input features. The SVM algorithm aims to minimize the following objective function:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b)) \tag{24}$$

Here, the regularization value C strikes a compromise between decreasing classification errors and optimizing the margin.

2) RANDOM FOREST

Similarly, Random Forest is one of the learning based techniques where a large number of models are generated in the training phase using different types of decision trees and then combined in an attempt to arrive at a decision. It borrows from the bootstrap technique whereby many models are developed at a time but with random data samples in order to improve on their efficiency as well as reliability.

In a RF model, for each individual tree we use a random and bootstrap sample of the original data set with replacement. Moreover, in the process of growing each node, only a limited number of independent variables is tested for each node. This kind of randomness is useful in order to prevent over fitting and improves the model prediction accuracy. Simplified to clarify that RF aggregates predictions by averaging for regression and voting for classification.

The prediction from a RF can be mathematically represented as:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \tag{25}$$

where T is the number of trees in the forest, and $h_t(x)$ is the prediction of the t-th tree for the input x.

3) LOGISTIC REGRESSION

Logistic Regression is a model used in statistics for a binary configuration problem. It is also called a linear regression with respect to two or more x variables and calculates the chances of occurrence of an event having one or more predictor variables. The Logistic Regression model takes the output from the linear form of features and scales it between $0-1$ which provides a probability score. The logistic function is defined as:

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{26}$$

where z is a linear combination of the input features. For Logistic Regression, z is given by:

$$z = w^T x + b \tag{27}$$

Here, w is the weight vector, x is the input feature vector, and b is the bias term. The output is a probability value between 0 and 1, indicating the likelihood that the input belongs to the positive class.

The goal of LR is to find the parameters w and b that minimize the logistic loss (or cross-entropy loss):

$$L(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{28}$$

where y_i is the actual label, and $\hat{y}_i$ is the predicted probability for the i-th observation.

4) K-NEAREST NEIGHBOR ALGORITHM

K-Nearest Neighbor is a technique doing both the classification and regression and the method is simple and non-parametric. It operates on the principle that 'like produces like' especially when it comes to data inputs and outcomes. KNN assigns the output of a function based on the measure of distance between the query point and the training data set where the output is obtained by aggregative function of the K nearest neighbors.

Euclidean distance is also used to measure the distance between the two points whereas other distance measurements like Manhattan distance can also be used also. This is true particularly with regards to k, the number of neighbors selected, since this determines the performance of the model. If the value is small, it leads to overfitting of the data, whereas, if the value of k is large then under fitting of data occurs.

The Euclidean distance between two points x and x' in an n-dimensional space is calculated as:

$$d(x, x') = \sqrt{\sum_{i=1}^{n}(x_i - x'_i)^2} \tag{29}$$

After calculating the distances to all training points, the KNN algorithm selects the k nearest neighbors and predicts the label based on majority voting (for classification) or averaging the output values (for regression).

E. PERFORMANCE METRICS

It is absolutely necessary to measure the performance of models developed using machine learning, and that's where performance metrics come into play. These are numerical values that help in evaluating how well a model is performing on a given dataset. This evaluation assists in selecting the correct model and optimizing the relevant hyperparameters. In the subsequent sub-sections, we explain some of the most commonly used performance measures, including Accuracy, Loss, F_1-Score, Precision, Recall, and Confusion Matrix.

1) ACCURACY

Accuracy is a straightforward and frequently used metric in classification scenarios. It determines the ratio of correct predictions made within the dataset relative to the total sample size of the dataset. Mathematically, accuracy is expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{30}$$

where:

- *TP* (True Positives) indicates the count of correctly positive predictions.
- *TN* (True Negatives) indicates the count of correctly negative predictions.
- *FP* (False Positives) indicates the number of incorrectly negative predictions that were wrongly taken as positive.
- *FN* (False Negatives) indicates the number of incorrectly positive predictions that were wrongly taken as negative.

2) LOSS

Loss functions measure the divergence between a predefined goal and the actual outcome in a given dataset. These functions are particularly useful during the training phase of machine learning models, especially within supervised learning, where the primary objective is to minimize the loss function to improve the model's accuracy. Various loss functions are used depending on the specific task at hand:

Mean Squared Error (MSE): This is calculated as the average of the squared differences between the expected and actual values, and it is commonly used in regression tasks.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{31}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and n is the number of instances.

CEL, or Cross-Entropy Loss which is frequently employed in classification problems, especially in logistic regression and neural networks, calculates the discrepancy between the actual label and the anticipated probability:

$$\text{CEL} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{32}$$

where y_i is the actual binary label, and $\hat{y}_i$ is the predicted probability.

3) PRECISION

Precision is a metric that evaluates the accuracy of a model's positive predictions. It is calculated by taking the number of true positives and dividing it by the total number of positive predictions, which includes both true positives and false positives. The formula for precision is:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{33}$$

Precision is particularly valuable in situations where the cost of false positives is high, as it focuses on the accuracy of positive predictions.

4) RECALL

Recall, often referred to as the True Positive Rate or Sensitivity, measures the model's ability to correctly identify all positive cases. It is computed by taking the ratio of true positives to the sum of true positives and false negatives. Recall is defined as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{34}$$

Recall is crucial in contexts where the consequences of missing a positive instance (i.e., a false negative) are severe.

5) F_1 SCORE

The F_1 Score represents the harmonic mean of Precision and Recall, offering a balanced metric that takes both into account. It is particularly useful in situations with imbalanced class distributions or when both precision and recall are critical. The F_1 Score, which ranges between 0 and 1, is calculated as follows:

$$F_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{35}$$

6) CONFUSION MATRIX

The Confusion Matrix is a detailed tabular representation that summarizes the performance of a classification model. It provides the counts of true positives, true negatives, false positives, and false negatives, allowing for a comprehensive evaluation of the model's performance.

For a binary classification problem, a typical confusion matrix is arranged as follows:

	Predicted Positive	Predicted Negative
Actual Positive	*TP*	*FN*
Actual Negative	*FP*	*TN*

(36)

The Confusion Matrix is highly useful because it not only provides insights into the model's overall accuracy but also highlights specific types of errors, which is critical for refining model performance.

IV. RESULTS AND DISCUSSION

As outlined in the Methodology section, the evaluation process begins with a performance comparison between two datasets—the original and the augmented dataset—and among four deep learning models: DenseNet161, ResNet152, MobileNetV3 Large, and MobileNetV3 Small. This comparison is conducted to determine the most suitable dataset and deep learning model for the feature extraction phase of the proposed integrated learning approach. Subsequently, the extracted features are subjected to selection by three metaheuristic feature selection algorithms: Whale Optimization Algorithm, Salp Swarm Algorithm, and Harris Hawks Optimization. The objective of this step is to identify the most suitable algorithm for selecting significant features and filtering out irrelevant ones. Finally, the selected features are classified using the four machine learning models mentioned above to determine the classifier that achieves the highest performance. Detailed results and discussions for each step are presented below.

A. EVALUATION OF DEEP LEARNING MODELS USING ORIGINAL AND AUGMENTED DATASETS

This phase of the evaluation encompasses the assessment of accuracy and loss for both the training and testing datasets over the course of 50 epochs, for both the original and augmented datasets. The results are illustrated in Figures 7, 8, 9, and 10.

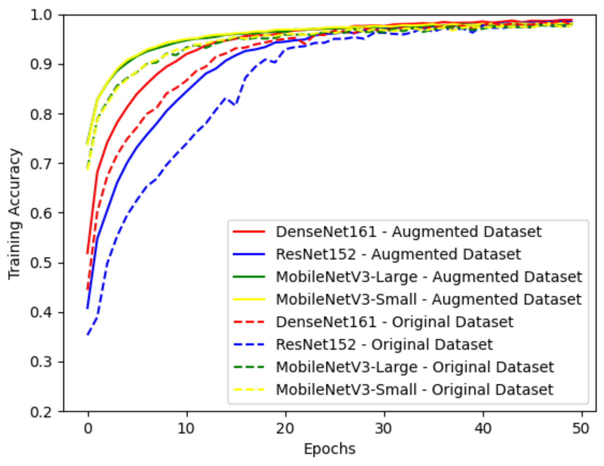

FIGURE 7. Training accuracy over epochs for deep learning models.

The training accuracy of the eight models (derived from the two dataset types applied to the four deep learning models) is depicted in Figure 7. The results indicate that accuracy levels tend to saturate at approximately 99%. DenseNet161 model has the highest training accuracy that is equal 98.85% when using the augmented dataset, while the lowest accuracy is recorded at 97.74% for MobileNetV3-Large using the original dataset, from the 20th epoch onward.

In addition, when testing the deep learning models with the help of such augmented dataset, the testing accuracy level demonstrated in Figure 8 remains high performance. For instance, DenseNet161 achieves the highest accuracy at 96.3%. However, the best performance of models using

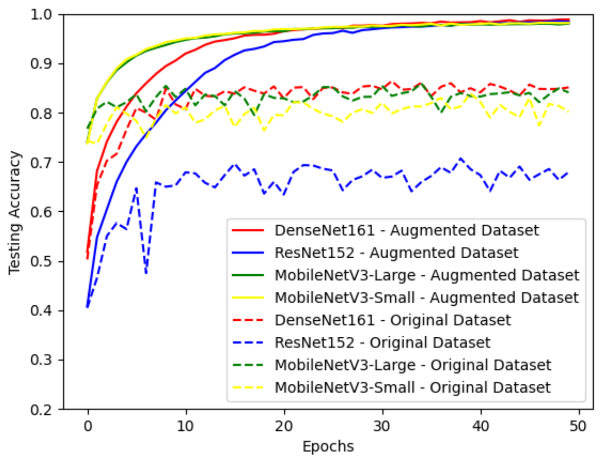

FIGURE 8. Testing accuracy over epochs for deep learning models.

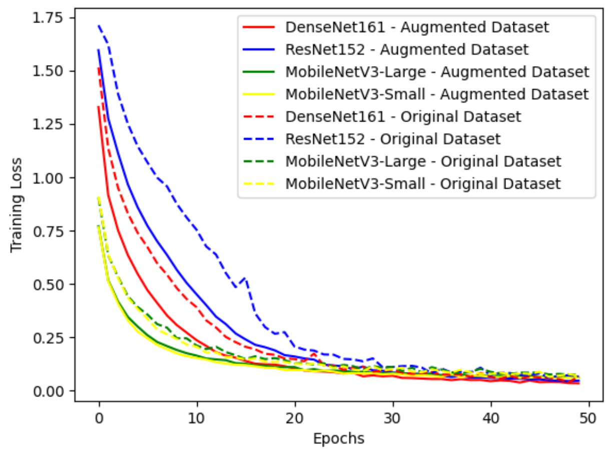

FIGURE 9. Training loss over epochs for deep learning models.

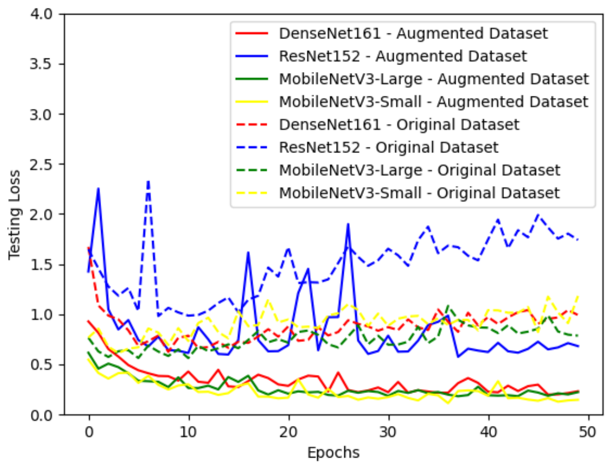

FIGURE 10. Testing loss over epochs for deep learning models.

our augmented datasets are much better as compared to original dataset. The ResNet152 model exhibits the poorest performance with an accuracy of 70.7% when testing in

TABLE 2. Performance evaluations of deep learning models using original and augmented dataset.

Dataset	DL Model	Highest testing accuracy	Precision	Recall	F_1-Score	Training time (per epoch)	Execution time (per image)
Original	DenseNet161	0.8638	0.8130	0.8110	0.8103	118.44s	0.0851s
	ResNet152	0.7070	0.6896	0.6796	0.6784	107.75s	0.0834s
	MobileNetV3-Large	0.8591	0.8360	0.8309	0.8305	85.32s	0.0498s
	MobileNetV3-Small	0.8087	0.7898	0.7791	0.7779	85.56s	0.0485s
Augmented	DenseNet161	0.9630	0.9532	0.9529	0.9529	388.32s	0.0665s
	ResNet152	0.8643	0.8490	0.8469	0.8466	356.93s	0.0531s
	MobileNetV3-Large	0.9553	0.9446	0.9445	0.9445	221.09s	0.0424s
	MobileNetV3-Small	0.9607	0.9516	0.9515	0.9515	201.63s	0.0575s

original dataset, while the other models achieve accuracy of 80.87%, 85.91%, and 86.38% for MobileNetV3-Small, MobileNetV3-Large, and DenseNet161, respectively.

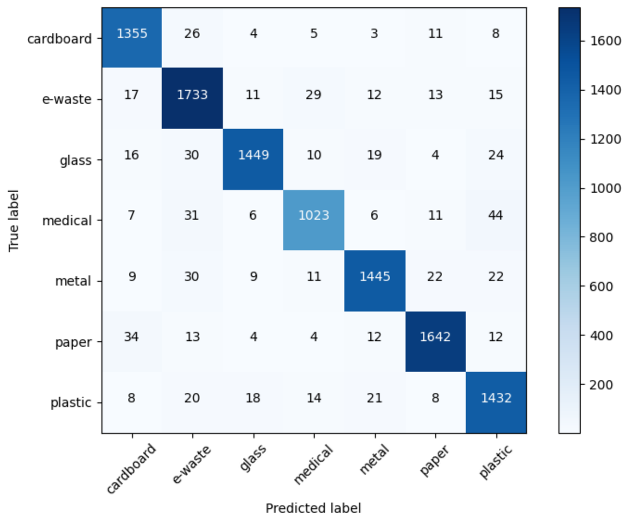

FIGURE 11. Confusion matrix of DenseNet161 using augmented dataset.

Likewise, the training and testing loss over epochs, as depicted in Figure 9, depict a good learning performance in terms of training accuracy, which have smooth curves and constant decreasing performance from epoch 20th and above (except for ResNet152 trained using the original dataset that saturates from epoch 30th). Smooth curves, as seen in Figure 10, demonstrate the testing loss results when models are trained on the augmented dataset. In contrast, fluctuations in the curve highlight the suboptimal performance of models trained solely on the original dataset.

Besides the aforementioned measures, other measures of performance shown in Table 2 are precision, recall, F_1-Score, time taken per epoch for training and time taken per image for execution. Further, we note that the time taken to train and execute DenseNet161 and ResNet152 models is comparatively higher than MobileNetV3 models, for each epoch as well as for each image. From the above results the augmented dataset has been selected as the main dataset and DenseNet161 has been selected as the major deep learning model for the feature extraction phase based on its high accuracy and reliability. The confusion matrix of DenseNet161 model, performed on the augmented samples of the testing folder containing 20% of the total 53,346 images is shown in the Figure 11.

B. EVALUATIONS OF INTEGRATED LEARNING MODELS

1) FEATURE REDUCTIONS

From the Global Average Pooling layer of the DenseNet161 model, a total of 2,208 features were extracted and subsequently selected using three metaheuristic feature selection algorithms.

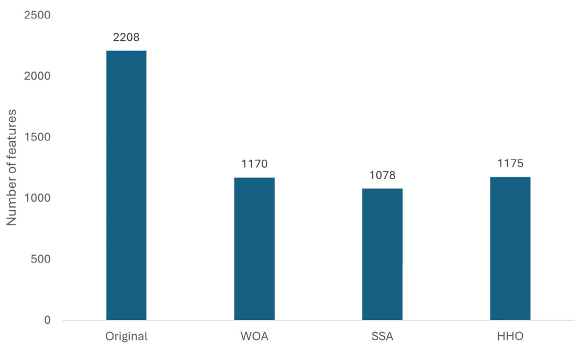

FIGURE 12. Number of original feature and selected feature.

As depicted in Figure 12 and detailed in Table 3, the WOA filtered out 1,038 non-essential features, thereby retaining 1,170 significant features—a reduction of 47.01% from the original set. The SSA achieved a reduction of 51.18%, retaining 1,078 important features. Lastly, the HHO algorithm selected 1,175 necessary features, filtering out 1,033 insignificant ones, which corresponds to a reduction of 46.78%.

2) PERFORMANCE EVALUATIONS

Table 3 presents the results of the performance evaluation comparing combination models without feature selection algorithms and integrated models with feature selection algorithms. The evaluations take into account Accuracy, Precision, Recall, F_1-Score value, time taken to select significant features in the datasets, time taken for each image, execution time per image, and the percentage of reduction in time.

The integrated model, which utilizes Harris Hawk Optimization for feature selection combined with a Support Vector Machine classifier, has demonstrated the most favorable outcomes and has been identified as the optimal approach. This model achieved the highest scores with

TABLE 3. Performance evaluations of integrated models.

Selection Algorithm	ML Model	Accuracy	Precision	Recall	F_1-Score	Selection time (per dataset)	Execution time (per image)
None	SVM	0.9238	0.9239	0.9238	0.9238	None	0.0036s
	RF	0.8673	0.8673	0.8673	0.8667	None	1.24e-05s
	LR	0.8925	0.8924	0.8925	0.8923	None	1.03e-05s
	KNN	0.8832	0.8838	0.8832	0.8828	None	4.27e-05s
WOA	SVM	0.9723	0.9723	0.9723	0.9723	22.02m	0.0028s (↓ 22.2%)
	RF	0.9402	0.9403	0.9402	0.9401	22.02m	9.5e-06s (↓ 23.4%)
	LR	0.9079	0.9080	0.9079	0.9079	22.02m	8.57e-06s (↓ 16.8%)
	KNN	0.9404	0.9409	0.9404	0.9404	22.02m	3.23e-05s (↓ 24.4%)
SSA	SVM	0.9740	0.9740	0.9740	0.9740	37.91m	0.0022s (↓ 38.9%)
	RF	0.9403	0.9405	0.9403	0.9402	37.91m	9.43e-06s (↓ 24.0%)
	LR	0.9135	0.9135	0.9135	0.9135	37.91m	8.52e-06s (↓ 17.3%)
	KNN	0.9396	0.9401	0.9396	0.9396	37.91m	3.16e-05s (↓ 26.0%)
HHO	**SVM**	**0.9745**	**0.9745**	**0.9745**	**0.9745**	**5.56m**	**0.0028s** (↓ 22.2%)
	RF	0.9369	0.9370	0.9369	0.9368	5.56m	9.43e-06s (↓ 24%)
	LR	0.9173	0.9174	0.9173	0.9173	5.56m	8.16e-06s (↓ 20.8%)
	KNN	0.9403	0.9407	0.9403	0.9402	5.56m	3.41e-05s (↓ 20.1%)

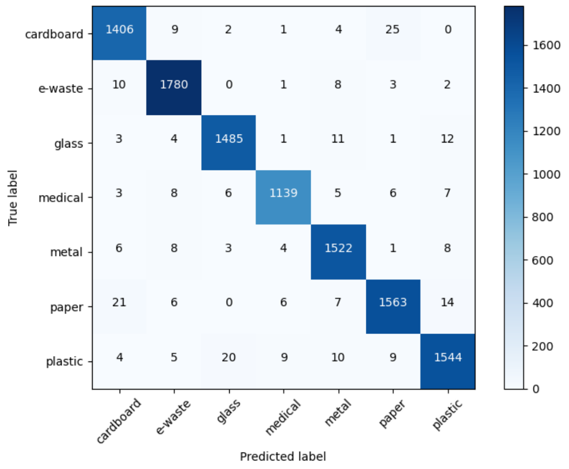

FIGURE 13. Confusion matrix of integrated learning method using DenseNet161 as feature extraction, HHO as feature selection and SVM as classification.

the performance of 97.45% for the first four evaluation parameters and the feature selection time of 5.56 minutes. Additionally, the reduction in the number of features led to a decrease in execution time, indicating that this approach requires fewer computational resources for implementation compared to models that do not employ a feature selection step. Although the classification time for SVM did not surpass that of other methods, 0.0028 seconds per image (approximately 357 frames per second) remains within acceptable limits. Finally, Figure 13 displays the confusion matrix for our primary integrated learning method, evaluated on the testing folder (20% of the augmented dataset).

C. COMPARE WITH KEY REFERENCE

As shown in Table 4, Ahmed Khan [18] introduced ten deep learning models for the TrashBox dataset, achieving testing accuracies ranging from 81.8% to 89.62%, with the ResNeXt-101 model yielding the highest performance in 2024.

In the same year, Das et al. [47] reported a higher testing accuracy of 95% by utilizing a DenseNet161 transfer learning model in combination with the TrashBox dataset and additional datasets. Similarly, in 2022, Rayhan and Rifai [48] demonstrated a testing accuracy of 93.1% using a DenseNet169 model, also through a combination of datasets that included TrashBox. These findings suggest that the proposed integrated learning approach in this research has the potential to surpass previous results due to the increased depth and complexity of its structure.

TABLE 4. The results of reference papers in same TrashBox dataset or its variants.

Year	Research	Dataset	Model	Testing Accuracy
-	**Our proposed method**	**Original TrashBox**	**Integrated Learning with Data Augmentation**	**0.9745**
2024	Yanfahmi Rayhan et al., [47]	TrashBox & other dataset	DenseNet121 Transfer learning model	0.95
2024	Haroon Ahmed Khan et al., [18]	Original TrashBox	ResNeXt-101	0.8962
2022	Dhrubajyoti Da et al., [48]	TrashBox & other dataset	DenseNet169	0.931

V. CONCLUSION

As a result, this research greatly benefits the field of Municipal Solid Waste (MSW) management, crucial for promoting environmental sustainability. The application of integrated learning models for waste categorization enhances sorting efficiency, minimizes contamination in recycling streams, and optimizes waste sorting technology. Automated identification and sorting improve recycling rates and reduce landfill volumes, mitigating the negative impacts of improper waste disposal. By supporting smart waste management through improved source-based sorting, this model advances environmental sustainability efforts. Thus, the presented methodology is not only a technical innovation but also a valuable tool for environmentally friendly waste management on a global scale.

Comprehensive assessments of deep learning models on original and augmented datasets indicated that DenseNet161 on augmented data achieved the highest overall accuracy and stability, making it the optimal choice for feature extraction in the proposed integrated learning scheme. Metaheuristic algorithms for feature selection further optimized the model by reducing the number of features while retaining the most relevant ones. Among the feature selection algorithms, Harris Hawk Optimization paired with a Support Vector Machine classifier yielded the best results, achieving the highest scores across all key metrics, including accuracy, precision, recall, and F_1-Score. Although SVM's execution time per image was slightly longer than other methods, it remained within acceptable limits, making the classifier both efficient and accurate. The confusion matrix results confirm the integrated method's effectiveness, demonstrating that this approach is suitable for high-performance classification tasks where both accuracy and response time are critical.

Compared to recent studies on the same dataset, this research achieved a notable improvement in accuracy, reaching 97.45% compared to previous results of 95% and 89.62%. This enhancement can be attributed to the proposed three-stage approach: deep learning model extraction, metaheuristic-based feature selection, and machine learning classification. Our DenseNet161-HHO-SVM approach emphasizes that dataset augmentation, careful model selection, and feature optimization are vital for developing models with strong generalization abilities.

Future work will focus on deploying this integrated model on an embedded GPU-based device, such as the Jetson Nano, to enable high-precision solid waste classification in real-time scenarios. Since training occurs offline, there is no impact on real-time performance, and computational requirements for training are comparable to other algorithms, making this approach feasible. Applying DenseNet161 on an Edge-AI device optimized with TensorRT is expected to improve execution speed, enabling real-time performance despite the model's higher computational load [19]. DenseNet161 was chosen for its accuracy and feature extraction strengths over lighter models, such as MobileNetV3, making it a robust choice for applications requiring dependable classification performance.

In resource-constrained settings, such trade-offs between accuracy and efficiency become crucial. The next stage of this research will involve developing mathematical models to minimize execution time while enhancing accuracy [49], [50]. These optimizations will improve the system's efficiency on embedded platforms to meet real-time response criteria. Our aim is to reduce computational complexity while retaining the accuracy of DenseNet161, ensuring the model's suitability for efficient solid waste classification on the Jetson Nano, which requires high accuracy despite its limited computational power.

ACKNOWLEDGMENT

This work has been conducted through the RMIT Vietnam Research Grants-2024, administered by the Office for Research & Innovation.

REFERENCES

[1] J. Gutberlet and S. M. N. Uddin, "Household waste and health risks affecting waste pickers and the environment in low- and middle-income countries," *Int. J. Occupational Environ. Health*, vol. 23, no. 4, pp. 299–310, Oct. 2017.

[2] *Municipal Waste Generation Worldwide As of 2022, By Select Country*, Statista, Hamburg, Germany, 2024.

[3] I. Ismiyati, I. Purnawan, and M. Kadarisman, "Effectiveness of environmental management based on trash in the city of Depok," *MATEC Web Conf.*, vol. 58, Jan. 2016, Art. no. 01008.

[4] M. A. Shahen, "How development sectors are contributing to waste management in Bangladesh," *Amer. J. Environ. Econ.*, vol. 3, no. 1, pp. 59–66, Jul. 2024.

[5] S. A. Urme, "Dhaka landfill waste practices: addressing urban pollution and health hazards," *MATEC Web Conf.*, vol. 2, no. 1, pp. 700–716, 2021.

[6] M. Hoang, T. Fujiwara, and S. T. Pham Phu, "Municipal solid waste characterisation and waste management issues in a tourist City–Hoi An, Vietnam," *J. JSCE*, vol. 5, no. 1, pp. 123–132, 2017.

[7] *Trends in Solid Waste Management*, World Bank, Washington, DC, USA, 2024.

[8] J. Bogner, R. Pipatti, S. Hashimoto, C. Diaz, K. Mareckova, L. Diaz, P. Kjeldsen, S. Monni, A. Faaij, Q. Gao, T. Zhang, M. Abdelrafie Ahmed, R. T. M. Sutamihardja, and R. Gregory, "Mitigation of global greenhouse gas emissions from waste: Conclusions and strategies from the intergovernmental panel on climate change (IPCC) fourth assessment report. Working group III (Mitigation)," *Waste Manage. Res., J. Sustain. Circular Economy*, vol. 26, no. 1, pp. 11–32, Feb. 2008.

[9] L. Giusti, "A review of waste management practices and their impact on human health," *Waste Manage.*, vol. 29, no. 8, pp. 2227–2239, Aug. 2009.

[10] Y. Yao, C. Ramu, A. Procher, J. Littlejohns, J. M. Hill, and J. W. Butler, "Potential for thermo-chemical conversion of solid waste in Canada to fuel, heat, and electricity," *Waste*, vol. 1, no. 3, pp. 689–710, Aug. 2023.

[11] A. Kumar and S. R. Samadder, "A review on technological options of waste to energy for effective management of municipal solid waste," *Waste Manage.*, vol. 69, pp. 407–422, Nov. 2017.

[12] A. K. Awasthi and J. Li, "Management of electrical and electronic waste: A comparative evaluation of China and India," *Renew. Sustain. Energy Rev.*, vol. 76, pp. 434–447, Sep. 2017.

[13] V. Forti, C. Baldé, R. Kuehr, and G. Bel, *The Global E-Waste Monitor 2020. Quantities, Flows, and the Circular Economy Potential*, 2020. [Online]. Available: https://ewastemonitor.info/wp-content/uploads/2020/11/GEM_2020_def_july1_low.pdf

[14] M. R. Haque, F. N. Chowdhury, A. Hossain, R. Akter, and M. M. Rahman, "An emerging concern of medical waste management in Rohingya refugee camps at Cox's bazar, Bangladesh: Existing practice and alternatives," *Frontiers Environ. Sci.*, vol. 11, Jul. 2023, Art. no. 1149314.

[15] B.-K. Lee, M. J. Ellenbecker, and R. Moure-Ersaso, "Alternatives for treatment and disposal cost reduction of regulated medical wastes," *Waste Manage.*, vol. 24, no. 2, pp. 143–151, 2004.

[16] J. A. S. Nelson L. Nemerow, and F. J. Agardy, *Environmental Engineering*, 6th ed. Wiley, 2009.

[17] D. Khan and S. R. Samadder, "Municipal solid waste management using geographical information system aided methods: A mini review," *Waste Manage. Research: J. Sustain. Circular Economy*, vol. 32, no. 11, pp. 1049–1062, Nov. 2014.

[18] H. A. Khan, S. S. Naqvi, A. A. K. Alharbi, S. Alotaibi, and M. Alkhathami, "Enhancing trash classification in smart cities using federated deep learning," *Sci. Rep.*, vol. 14, no. 1, May 2024, Art. no. 11816.

[19] H. T. Minh, L. Mai, and T. V. Minh, "Performance evaluation of deep learning models on embedded platform for edge AI-based real time traffic tracking and detecting applications," in *Proc. 15th Int. Conf. Adv. Comput. Appl. (ACOMP)*, Nov. 2021, pp. 128–135.

[20] L. V. Tran, H. M. Tran, T. M. Le, T. T. M. Huynh, H. T. Tran, and S. V. T. Dao, "Application of machine learning in epileptic seizure detection," *Diagnostics*, vol. 12, no. 11, p. 2879, Nov. 2022.

[21] S. V. T. Dao, Z. Yu, L. V. Tran, P. N. K. Phan, T. T. M. Huynh, and T. M. Le, "An analysis of vocal features for Parkinson's disease classification using evolutionary algorithms," *Diagnostics*, vol. 12, no. 8, p. 1980, Aug. 2022.

[22] Q. N. X. Phan, T. M. Le, H. M. Tran, L. V. Tran, and S. V. T. Dao, "Novel machine learning techniques for classification of rolling bearings," *IEEE Access*, early access, Jul. 19, 2024, doi: 10.1109/ACCESS.2024.3431040.

[23] L. Tonthat, V. T. Son Dao, H. T. Minh Tri, and M. T. Le, "A feature subset selection approach for predicting smoking behaviours," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, Jul. 2023, pp. 145–149.

[24] A. Masand, S. Chauhan, M. Jangid, R. Kumar, and S. Roy, "ScrapNet: An efficient approach to trash classification," *IEEE Access*, vol. 9, pp. 130947–130958, 2021.

[25] C. Bircanoglu, M. Atay, F. Beser, Ö. Genç, and M. A. Kizrak, "RecycleNet: Intelligent waste sorting using deep neural networks," in *Proc. Innov. Intell. Syst. Appl. (INISTA)*, Jul. 2018, pp. 1–7.

[26] O. Adedeji and Z. Wang, "Intelligent waste classification system using deep learning convolutional neural network," *Proc. Manuf.*, vol. 35, pp. 607–612, Jan. 2019.

[27] Y. Chu, C. Huang, X. Xie, B. Tan, S. Kamal, and X. Xiong, "Multilayer hybrid deep-learning method for waste classification and recycling," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–9, Nov. 2018.

[28] H. Zhou, X. Yu, A. Alhaskawi, Y. Dong, Z. Wang, Q. Jin, X. Hu, Z. Liu, V. G. Kota, M. Abdulla, S. Ezzi, B. Qi, J. Li, B. Wang, J. Fang, and H. lu, "A deep learning approach for medical waste classification," *Sci. Rep.*, vol. 12, no. 1, p. 2159, 2022.

[29] A. H. Vo, L. Hoang Son, M. T. Vo, and T. Le, "A novel framework for trash classification using deep transfer learning," *IEEE Access*, vol. 7, pp. 178631–178639, 2019.

[30] Z. Yang, Y. Bao, Y. Liu, Q. Zhao, H. Zheng, and Y. Bao, "Research on deep learning garbage classification system based on fusion of image classification and object detection classification," *Math. Biosci. Eng.*, vol. 20, no. 3, pp. 4741–4759, 2022.

[31] M. Zeng, X. Lu, W. Xu, T. Zhou, and Y. Liu, "PublicGarbageNet: A deep learning framework for public garbage classification," in *Proc. 39th Chin. Control Conf. (CCC)*, Jul. 2020, pp. 7200–7205.

[32] B. Fu, S. Li, J. Wei, Q. Li, Q. Wang, and J. Tu, "A novel intelligent garbage classification system based on deep learning and an embedded Linux system," *IEEE Access*, vol. 9, pp. 131134–131146, 2021.

[33] W. Liu, H. Ouyang, Q. Liu, S. Cai, C. Wang, J. Xie, and W. Hu, "Image recognition for garbage classification based on transfer learning and model fusion," *Math. Problems Eng.*, vol. 2022, pp. 1–12, Aug. 2022.

[34] N. Kumsetty, A. Nekkare, S. S. Kamath, and M. A. Kumar, "TrashBox: Trash detection and classification using quantum transfer learning," in *Proc. 31st Conf. Open Innov. Assoc. (FRUCT)*, Helsinki, Finland, 2022, pp. 125–130, doi: 10.23919/FRUCT54823.2022.9770922.

[35] Thung and M. Yang, "TrashNet: Dataset of images of trash; Torch-based CNN for garbage image classification," GitHub Repository, 2016. [Online]. Available: https://github.com/garythung/trashnet

[36] P. F Proença and P. Simões, "TACO: Trash annotations in context for litter detection," 2020, *arXiv:2003.06975*.

[37] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[38] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[40] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[41] T. M. Le, L. V. Tran, and S. V. T. Dao, "A feature selection approach for fall detection using various machine learning classifiers," *IEEE Access*, vol. 9, pp. 115895–115908, 2021.

[42] T. M. Le, T. N. Pham, and S. V. T. Dao, *Novel Wrapper-Based Feature Selection for Heart Failure Prediction Using an Adaptive Particle Swarm Grey Wolf Optimization*. Cham, Switzerland: Springer, 2021, pp. 315–336.

[43] I. Guyon and A. Elisseeff, "An introduction of variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jun. 2003.

[44] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[45] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[46] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.

[47] D. Das, A. Sen, S. M. M. Hossain, and K. Deb, "Trash image classification using transfer learning based deep neural network," in *Intelligent Computing & Optimization*, P. Vasant, G.-W. Weber, J. A. Marmolejo-Saucedo, E. Munapo, and J. J. Thomas, Eds., Cham, Switzerland: Springer, 2023, pp. 561–571.

[48] Y. Rayhan and A. Pratama Rifai, "Multi-class waste classification using convolutional neural network," *Appl. Environ. Res.*, vol. 46, no. 2, 2024. [Online]. Available: https://doi.org/10.35762/AER.2024021

[49] H. M. Tran, K. T. Pham, T. M. Vo, T.-H. Le, T. T. M. Huynh, and S. V. T. Dao, "A new approach for estimation of physical properties of irregular shape fruit," *IEEE Access*, vol. 11, pp. 46550–46560, 2023.

[50] H. M. Tran, K. T. Pham, T. M. Vo, L. Tonthat, T. T. M. Huynh, and S. V. T. Dao, "Physical characteristics estimation for irregularly shaped fruit using two cameras," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, Jul. 2023, pp. 165–169.

HIEU M. TRAN received the Bachelor of Engineering degree in automation and control engineering and the Master of Engineering degree in electronics engineering from the International University, Vietnam National University Ho Chi Minh City, Vietnam, in 2021 and 2023, respectively. His research interests include artificial intelligence on the edge, deep learning, and optimization.

TUAN M. LE received the bachelor's degree in electrical engineering from International University, Vietnam National University, Vietnam, in 2018, and the M.Sc. degree in electronics engineering, in 2022. His research interests include embedded systems the Internet of Things (IoT) and machine learning (ML).

HUNG V. PHAM received the master's degree from RMIT VN as the first scholarship cohort when ECE Program was deployed in VN, in 2011, and Ph.D. degree in semiconductor physics from RMIT University, in 2019. He became an Engineer with Intel VN working on Test and Assembly Process and helped to bring the first CPU production line to Intel VN. Since then, he works with RMIT University Vietnam.

MINH T. VU is currently an Associate Lecturer with RMIT University Vietnam. His research interests include software development, computer vision, and machine learning.

SON VU TRUONG DAO received the B.Eng. degree in aeronautical engineering, the M.Sc. degree in manufacturing systems and technology, and the Ph.D. degree in sensors technology, in 2004, 2005, and 2010, respectively. From 2006 to 2008, he was with Hylax Ltd., Singapore, designing high power laser systems. From 2010 to 2012, he was a Postdoctoral Fellow with Kindai University, Japan. From 2012 to 2015, he was a Senior Scientist with Ritsumeikan University, Japan. His research interests include the development of advanced imaging sensors and systems and applied artificial intelligence.