sustainability

MDPI

Article

R3sNet: Optimized Residual Neural Network Architecture for the Classification of Urban Solid Waste via Images

Mirna Castro-Bello [1], V. M. Romero-Juárez [1], J. Fuentes-Pacheco [2,*], Cornelio Morales-Morales [3], Carlos V. Marmolejo-Vega [1,*], Sergio R. Zagal-Barrera [1], D. E. Gutiérrez-Valencia [1] and Carlos Marmolejo-Duarte [4]

[1] National Technological Institute of Mexico, Technological Institute of Chilpancingo, Chilpancingo de los Bravo 39090, México; mirna.cb@chilpancingo.tecnm.mx (M.C.-B.); mg16520276@chilpancingo.tecnm.mx (V.M.R.-J.); sa@chilpancingo.tecnm.mx (S.R.Z.-B.); diego.gv@chilpancingo.tecnm.mx (D.E.G.-V.)
[2] National Technological Institute of Mexico, National Center for Research and Technological Development, Cuernavaca 62490, Morelos, Mexico
[3] National Technological Institute of Mexico, Technological Institute of San Juan del Río, San Juan del Río Querétaro 76800, México; cornelio.mm@chilpancingo.tecnm.mx
[4] Center of Land Policy and Valuations, Barcelona School of Architecture (ETSAB), Polytechnic University of Catalonia, 08034 Barcelona, Spain; carlos.marmolejo@upc.edu
* Correspondence: jorge.fp@cenidet.tecnm.mx (J.F.-P.); carlos.mv@chilpancingo.tecnm.mx (C.V.M.-V.); Tel.: +52-7772328308 (J.F.-P.); +52-7472555547 (C.V.M.-V.)

Abstract: Municipal solid waste (MSW) accumulation is a critical global challenge for society and governments, impacting environmental and social sustainability. Efficient separation of MSW is essential for resource recovery and advancing sustainable urban management practices. However, manual classification remains a slow and inefficient practice. In response, advances in artificial intelligence, particularly in machine learning, offer more precise and efficient alternative solutions to optimize this process. This research presents the development of a light deep neural network called R3sNet (three "Rs" for Reduce, Reuse, and Recycle) with residual modules trained end-to-end for the binary classification of MSW, with the capability for faster inference. The results indicate that the combination of processing techniques, optimized architecture, and training strategies contributes to an accuracy of 87% for organic waste and 94% for inorganic waste. R3sNet outperforms the pre-trained ResNet50 model by up to 6% in the classification of both organic and inorganic MSW, while also reducing the number of hyperparameters by 98.60% and GFLOPS by 65.17% compared to ResNet50. R3sNet contributes to sustainability by improving the waste separation processes, facilitating higher recycling rates, reducing landfill dependency, and promoting a circular economy. The model's optimized computational requirements also translate into lower energy consumption during inference, making it well-suited for deployment in resource-constrained devices in smart urban environments. These advancements support the following Sustainable Development Goals (SDGs): SDG 11: Sustainable Cities and Communities, SDG 12: Responsible Consumption and Production, and SDG 13: Climate Action.

Keywords: deep learning; waste classification; convolution neuronal network; optimized architecture; residual networks

Academic Editor: Jose Navarro Pedreño

Received: 8 March 2025
Revised: 8 April 2025
Accepted: 10 April 2025
Published: 14 April 2025

Citation: Castro-Bello, M.; Romero-Juárez, V.M.; Fuentes-Pacheco, J.; Morales-Morales, C.; Marmolejo-Vega, C.V.; Zagal-Barrera, S.R.; Gutiérrez-Valencia, D.E.; Marmolejo-Duarte, C. R3sNet: Optimized Residual Neural Network Architecture for the Classification of Urban Solid Waste via Images. *Sustainability* **2025**, *17*, 3502. https://doi.org/10.3390/su17083502

1. Introduction

In recent years, industrialization and urbanization have grown considerably, generating large amounts of municipal solid waste (MSW) and making it necessary to improve

sorting tasks. For example, in 2020, approximately 2010 million tons were generated; the World Bank forecasts that by 2030, this will increase to 2590 million tons and 3400 million tons by 2050. Of these, approximately 40% are deposited in landfills, 19% are recycled, and 11% are treated through incineration [1–3]. The efficient separation of MSW is key to promoting environmental sustainability and social well-being. Improving recycling processes reduces the ecological footprint, minimizes landfill use, and promotes the circular economy, aligning with global strategies such as the Sustainable Development Goals (SDGs). This research contributes to SDGs 11, 12, and 13, providing a technological solution to improve MSW management and contribute to a sustainable future.

Recycling tasks generally require high labor costs, so the recognition and automatic detection of waste through images replace manual sorting, thanks to advances in artificial intelligence (AI) [4]. Moreover, machine learning algorithms aimed at improving precision in automatic sorting can address complex challenges in recognition and classification. Convolutional neural networks (CNNs) comprise convolutional layers responsible for feature extraction and a fully connected layer that acts as a classifier, containing many neurons, making them a predominant methodology for image classification [4].

Various studies have used deep learning (DL) models to classify MSW. For example, in [5], a CNN for waste classification was developed using DenseNet121 and a genetic algorithm to optimize the fully connected layer, achieving 99.60% accuracy with a network of 8 million parameters. In [6], a waste image classification model based on DenseNet169 and a transfer learning strategy are presented, constructing a dataset featuring varied backgrounds (NWNU-TRASH) and a balanced distribution. A 70:30 data split was used for training and testing, reaching an accuracy of over 82%.

In [7], a hybrid multilayer convolutional neural network with few parameters was developed using 64 × 64-pixel images obtained from TrashNet, achieving an accuracy of 92.6%. In [3], a recyclable waste image classification model is proposed that applies image augmentation techniques based on an 18-layer architecture (ResNet18). Their proposal addresses the challenge of identifying and classifying images that present diverse characteristics by optimizing the original structure of the residual network, effectively integrating the most relevant features from the channel maps and compressing them within the spatial dimension, allowing for a more concise and efficient representation of visual information with an accuracy of 95.87%. In [8], a convolutional neural network inspired by VGGNet was proposed for waste classification. The MLH-CNN was optimized with fewer parameters and hybrid modules, and TrashNet was used as the dataset, achieving an accuracy of 92.6%. In [9], the authors proposed an Eco Cycle Classifying Deep Neural Network (ECCDN-Net) model for binary classification (organic and inorganic) of waste images using components from Densenet201 and Resnet18. They used 24,705 images, achieving a classification accuracy of 96.10% [10]. They compared the DenseNet121, MobileNet, ResNet50, and Xception architectures in the TrashNet dataset, concluding that DenseNet121 performs better when fine-tuning, with a precision rate of 95%. In [11], the authors present a VGG16 architecture to classify different types of garbage, trained with the TrashNet dataset. They achieved an accuracy of over 96%. On the other hand, in [12], a CNN model is proposed for classifying waste. Specifically, the Inception-v3 model was trained with a dataset obtained from different sources, achieving a classification accuracy of 92.5%.

Additionally, in [13], a model for recyclable waste classification using deep learning called RWNet was developed based on several ResNet structures; they used the TrashNet dataset with image augmentation techniques, achieving an accuracy of 88%. Finally, in [14], a small convolutional neural network was proposed for waste classification, featuring an adaptive image brightness algorithm to balance the background brightness during the

preprocessing phase. They also implemented a threshold replacement method designed to mitigate noise generated by shadows. They used the Canny operator to crop the white background and reduce visual interference, achieving an accuracy of 96.77% on their custom dataset and 93.72% on the TrashNet dataset. The binary classification of MSW offers advantages regarding recycling efficiency, cost reduction, and promotion of a circular economy. It is effective in scenarios with limited resources, low levels of environmental education, or high volumes of organic waste. Conversely, in situations where waste generation is low, infrastructure is restricted, or management objectives are primarily basic, it is not necessary to implement multi-class classification systems, as binary classification can be an efficient and effective solution.

In particular, the binary classification of MSW offers advantages regarding recycling efficiency, cost reduction, and promoting a circular economy. It is functional in scenarios with limited resources, low levels of environmental education, or high volumes of organic waste. On the other hand, where waste generation is low, there is limited infrastructure, or the management objectives are primarily essential, it is not necessary to implement multi-class classification systems because binary classification can be an efficient and effective solution. In this sense, binary classification represents an effective and robust strategy, providing a critical basis in real MSW management systems that could later be expanded to more detailed hierarchical schemes.

Unlike previous works that use basic convolutional architectures that process information sequentially, such as those in [5,6], this paper proposes an optimized residual architecture. Residual networks introduce skip connections that add the input of one layer to its output, preserving essential features of previous layers and thus preventing the gradients from diminishing too much (gradient vanishing problem). Using residual modules allows for designing deeper neural networks, capturing more complex patterns that are useful for improving their performance. On the other hand, while data from different studies present variability in backgrounds and resolutions, the dataset used here seeks to simplify visual characteristics to focus on learning the intrinsic properties of waste.

This research presents the development of an optimized architecture based on residual networks trained end-to-end for the binary classification of MSW, which can perform faster inference.

2. Materials and Methods

2.1. Creation of the Dataset

Data augmentation is an effective strategy for training classification models, particularly for object recognition. This is because high-dimensional images present various variation factors, many of which can be easily replicated [15].

Multiple public databases were consulted for the composition of the waste dataset, such as TrashNet [16], Garbage_Huawei [17], Waste Classification data [18], Garbage Dataset [19], Garbage Classification [20] and Garbage Pictures for Classification [21], as shown in Figure 1. The images in this dataset contain the waste items against a controlled background. This distinctive dataset allows the neural network to focus solely on the inherent features of the waste items, ensuring no bias is caused by the complex backgrounds against which the waste items are placed. A waste classification system trained with images against controlled backgrounds is particularly beneficial on conveyor belts with constant lighting and uniform backgrounds, as well as recycling machines where users manually place their waste items.

Figure 1. Sample of images from the six datasets used: (**a**) TrashNet, (**b**) Garbage Huawei, (**c**) Garbage Classification Data, (**d**) Garbage Dataset, (**e**) Garbage Classification, and (**f**) Garbage Pictures for Classification.

Each image was inspected to eliminate duplicates and ensure no repetitions across the sources. Content-based hash techniques were used to identify and filter similar images. Additionally, the images were carefully labeled into organic and inorganic categories.

The final dataset includes 3208 images, divided into 1574 images for the organic waste class and 1634 images for the inorganic waste class, with a proportional balance between both categories, as shown in Table 1 and Figure 2.

Table 1. Source information and number of images for each dataset.

Dataset Name	Link	Number of Images
TrashNet	https://github.com/garythung/trashnet, accessed on 21 December 2024	993
Garbage_HuaWei	https://www.kaggle.com/datasets/xiaohoua/garbage-huawei, accessed on 10 December 2024	548
Waste Classification data	https://www.kaggle.com/datasets/techsash/waste-classification-data, accessed on 21 December 2024	731
Garbage Dataset	https://www.kaggle.com/datasets/sumn2u/garbage-classification-v2, accessed on 10 December 2024	175
Garbage Classification (12 classes)	https://www.kaggle.com/datasets/mostafaabla/garbage-classification, accessed on 21 December 2024	275
Garbage Pictures for Classification	https://www.kaggle.com/datasets/mascot9183/garbage-pictures-for-classification, accessed on 10 December 2024	339

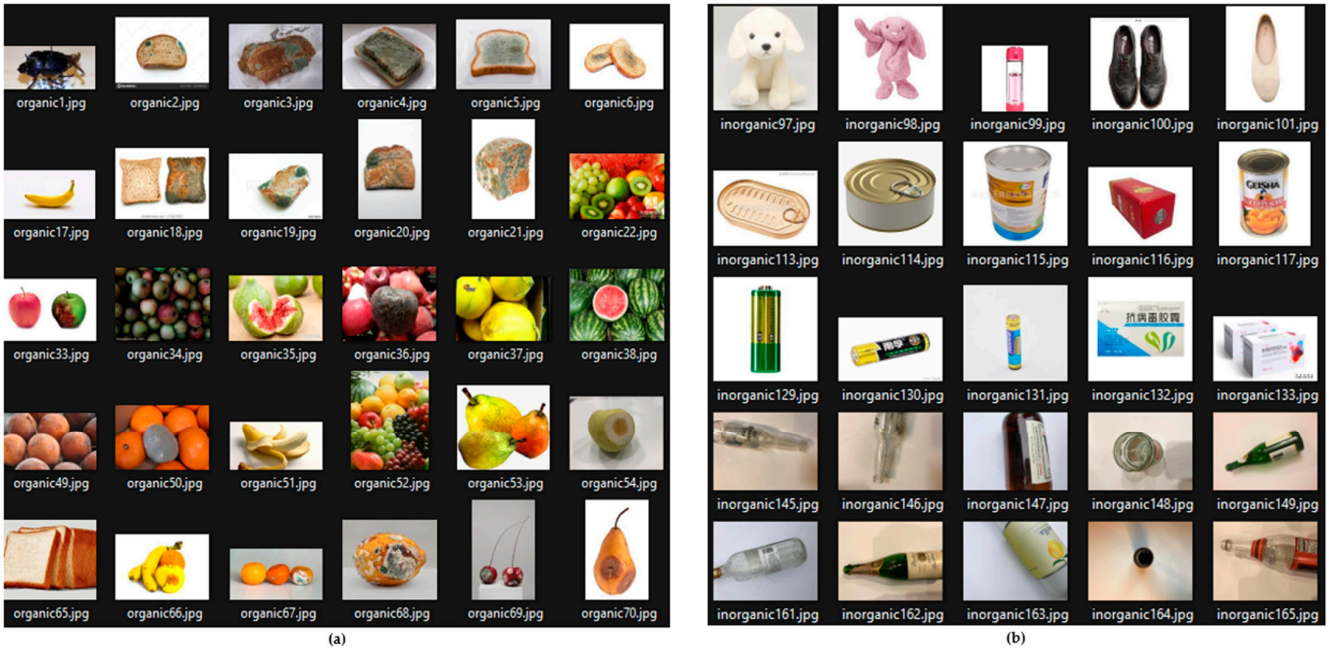

Figure 2. Samples from the dataset to evaluate the proposed network: (**a**) organic waste and (**b**) inorganic waste.

To ensure uniformity in processing, the images were converted to RGB format, regardless of their original format. This guaranteed that each image had three consistent channels, which are essential for processing by the neural network architecture. The RGB color space was chosen because it preserves the complete chromatic information of images, unlike grayscale, which discards color components. This preservation of the three channels (red, green, and blue) provides the network with additional features for classification, enabling it to identify textures, contours, and color patterns that are not perceptible in a monochromatic image. Consequently, RGB images offer variations in both light intensity and hue. Recent literature demonstrates that maintaining color can enhance classification accuracy; for instance, it has been reported that using the RGB scheme provides the most complete information to the model and results in superior performance compared to alternative spaces such as HSV, YCbCr, or grayscale images [18].

2.2. Data Augmentation

Image augmentation was performed online during network training using the ImageDataGenerator (https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (accessed on 8 March 2025)) class from the deep learning neural network library Keras [22]. The data augmentation techniques applied included rescaling, rotation, zoom, width shift, height shift, shear, horizontal flip, vertical flip, and fill mode; the ranges of these parameters are detailed in Table 2. The data distribution was 85% for training and 15% for validation, with a separate set of images used for testing. The test set comprised 569 images (both organic and inorganic), which were verified to ensure that none of these images were repeated or previously included in the training and validation sets. This careful separation guarantees that the performance reported for the R3sNet model is objective and genuinely representative of its actual generalization capacity, as the evaluation is carried out exclusively on data entirely new to the model.

Table 2. Values used for data augmentation techniques in the training phase.

ImageDataGenerator	
Parameter	**Value**
Rescale	1./255
Rotation	[0, 5]
Zoom	[0.0, 0.10]
Width shift	[0.0, 0.1]
Height shift	[0.0, 0.1]
Shear range	[0.0, 0.5]
Horizontal flip	True
Vertical flip	True
Fill mode	nearest

To feed the model with the processed images, data generators (flow_from_dataframe) were used, allowing images to be loaded directly from their paths. These generators were set up with a batch size of 16 images, a target size of 224 × 224 pixels, and binary classification mode for the labels. Additionally, the training generator used image randomization (shuffle = True), while the validation and test generators did not employ this option (shuffle = False), ensuring consistency in the model's evaluation.

This process ensured that the data used to train the model were balanced and efficiently prepared, improving overall performance and the system's generalization capacity.

The batch size hyperparameter was determined empirically by examining the maximum number of images, along with their weights and biases, that could fit into the available GPU VRAM memory for training. By considering more examples per batch, the gradient becomes more representative of the entire dataset, facilitating smoother convergence and reducing the number of iterations per epoch. In a small batch, the gradient is computed with fewer examples per iteration, resulting in a noisy gradient that can hinder training convergence.

2.3. R3sNet Network Proposal

ResNet was introduced in 2016 by [22], achieving first place in the ImageNet image classification challenge. However, models of this type require a lot of hardware resources, mainly because they are designed to solve a more complex problem. This leads to model size and inference time shortcomings when applied to specific tasks, such as classifying waste images [23]. These architectures' complexity and high number of parameters can be a disadvantage in environments with limited computational resources or when low-cost and efficient deployment is required.

The creation of R3sNet architecture is based on its ability to build deep neural networks without facing traditional problems like vanishing gradients and overfitting (see Figure 3). "R3" refers to waste management's three "Rs": Reduce, Reuse, and Recycle. Likewise, the allusion to ResNet, the base architecture with residual modules, is maintained. The ResNet introduced by [22] has proven highly effective in training deep networks, thanks to their skip connections that facilitate gradient flow during backpropagation. R3sNet is a modified and lighter version of the well-known ResNet18 architecture, explicitly designed to meet the needs of this study. By reducing the depth and adjusting the structure of the residual blocks, R3sNet maintains the computational efficiency and generalization capacity of ResNet18 while being optimized for binary classification tasks.

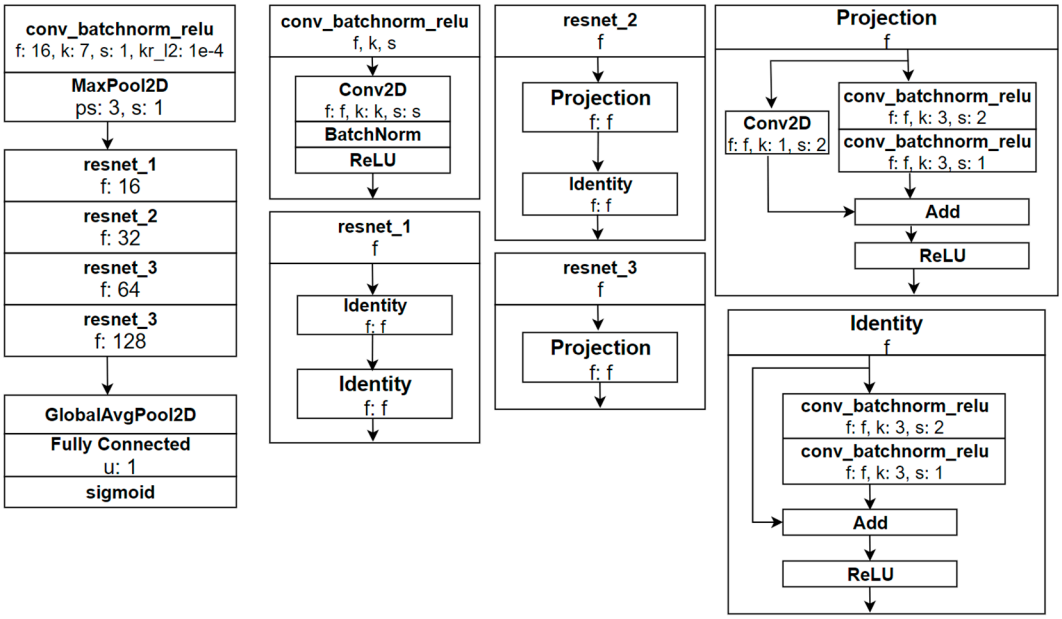

Figure 3. Diagram of R3sNet architecture proposed.

Inspired by variants such as ResNeXt [24] and lightweight architectures like MobileNetV2 [25], R3sNet incorporates modifications to reduce the number of parameters and lower computational resource consumption without significantly compromising performance in classifying organic and inorganic waste. These adaptations are crucial for deploying the model in resource-constrained environments and accelerating training and inference times. Additionally, by modifying the residual blocks to handle feature dimensions efficiently, R3sNet facilitates efficient information flow through the network, ensuring that discriminative features are preserved and effectively utilized for the final classification. This optimization is essential for tasks where differences between classes can be subtle and require precise feature extraction, as in the case of organic waste, where the extracted features include a more significant variability in textures and colors, reflecting the diversity of materials such as food remnants, leaves, and other biological components. On the other hand, inorganic waste usually exhibits smooth surfaces with more uniform patterns and reflections characteristic of materials like plastics, glass, and metals.

Although R3sNet is inspired by well-known architectures such as ResNet18, ResNeXt, and MobileNetV2, it incorporates specific modifications that optimize its computational efficiency and performance in particular tasks of MSW binary classification. Unlike ResNet18, our network significantly reduces the number of residual layers to minimize the number of parameters (from 11.7 M to 0.3 M). Unlike ResNeXt, neither grouped convolutions nor parallel paths were applied, but rather simple residual connections with ReLU activations and batch normalization were prioritized, further simplifying the structure. Finally, concerning

MobileNetV2, no inverted convolutional blocks with separable convolutions were used instead of standard residual blocks with classic 3 × 3 convolutions. These modifications were made with the explicit intention of obtaining an efficient, light, and fast model that could be deployed in devices with limited resources while maintaining high performance in accuracy and generalization capacity.

Below are the main characteristics of the architecture used (see Figure 3), including the kernel size k, the number of filters f, the kernel regularizer kr_l2, the pool size ps, and the stride s. Table 3 shows the number of parameters of R3sNet and the ResNet50 networks. R3sNet has only 0.3 M parameters, so it needs much less memory and storage space, facilitating its deployment in devices with limited hardware resources. Although R3sNet has more GFLOPS than ResNet18, it is still much lighter than ResNet50. R3sNet requires almost 1.5 times more operation than ResNet18, but the parameters are reduced by 97%. R3sNet maintains a manageable level of GFLOPS, allowing a balance between efficiency and computing capacity.

Table 3. The number of parameters of R3sNet, ResNet18, and ResNet50 with an input image size of 224 × 224 pixels.

Model	Parameters (M)	GFLOPS
R3sNet	0.3	2.70
ResNet18	11.7	1.81
ResNet50	25.6	7.75

- Input Layer

The input data consist of 224 × 224-pixel images with three color channels (RGB). These images are processed by the network through a series of convolutional layers that extract relevant features for the subsequent waste classification into organic and inorganic categories.

- Initial Convolutional Layer

The network begins with a convolutional layer that uses a 7 × 7 filter, a stride of 1, and 16 filters. This layer aims to extract initial spatial features from the input images. The two-dimensional convolution operation can be mathematically represented as follows:

$$Y[i,j] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X[i+m, j+n] \cdot W[m,n] + b \quad (1)$$

where Y[i,j] is the output value at position (i, j) after convolution, $X[i, j]$ is the input value at position (i, j), $W[m, n]$ is the filter value at position (m, n), b is the bias, and M and N are the filter's dimensions.

For the first convolutional layer, the ResNet approach of using 7 × 7 filter sizes, which capture larger and more global patterns, was retained. In the case of the images with residues used, filters with a larger receptive field in the first convolutional layer are incredibly beneficial since the input images are 224 × 224 (relatively large) and contain debris at different scales, capturing textures and edges of varying debris sizes. The 7 × 7 filters quickly reduce the spatial dimensions of the input, which decreases the computational cost for the following convolutional layers. Also, each filter has a total of 49 weights, allowing for a broader variety of initial patterns to be learned. Then, to achieve deeper layers with more channels, 3 × 3 filters are selected for the deeper convolutional layers. These filters facilitate the extraction of fine details to refine the classification and enable a controlled increase in the number of parameters.

- Batch Normalization

After convolution, batch normalization is applied before the activation functions of a layer to stabilize learning and accelerate model convergence. This operation adjusts the activations of each batch so that they have a mean near zero and a variance close to one, helping to mitigate the covariate shift problem. The operation is defined as follows:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2}$$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{4}$$

where x_i is the activation value for input i, μ_B and σ_B^2 are the mean and variance of the batch, respectively, ϵ is a small constant value to avoid division by zero, m is the batch size, and $\hat{x}_i$ is the normalized value. The normalization process during training is as follows: First, the batch mean and variance are calculated for each channel of the feature map using Formulas (3) and (4). Then, Formula (2) is used to normalize the activations, ensuring they have a mean of 0 and variance of 1. An additional transformation is applied to rescale and shift the normalized values with learnable parameters γ and β to obtain y_i using Formula (5). During inference, only cumulative statistics are used.

$$y_i = \gamma\, \hat{x}_i + \beta \tag{5}$$

- ReLU Activation Function

Finally, a ReLU activation function introduces non-linearity into the model, enabling the network to learn more complex data representations. It is defined as follows:

$$ReLU(x) = max(0, x) \tag{6}$$

This indicates that for any input x, the function returns to x if it is greater than 0, and 0 if x is less than or equal to 0. Mathematically, this can be expressed as follows:

$$ReLU(x) = \begin{cases} x\ if\ x > 0 \\ 0\ if\ x\ \leq 0 \end{cases} \tag{7}$$

This function is widely used in deep learning neural networks because it addresses the vanishing gradient problem, making the model easier to train and often achieving better performance. This sequence of operations (convolution, batch normalization, and ReLU activation) allows the network to extract and enhance significant features from the input images, facilitating the subsequent waste classification into organic and inorganic categories.

- Residual Blocks

After the initial block, the model comprises four residual blocks. The first block contains 16 filters, two repetitions, and one stride; the second block consists of 32 filters, two repetitions, and two strides; the third block consists of 64 filters, one repetition, and two strides; and the fourth block consists of 128 filters, one repetition, and two strides. These blocks enable the creation of deep networks without the issues associated with vanishing

gradients. A residual block in the R3sNet architecture is defined conditionally, depending on whether there is a change in the dimensionality of the features:

$$y = \begin{cases} ReLU(F(x, \{W_i\}) + x) & \text{if there is no change in dimensionality} \\ ReLU(F(x, \{W_i\}) + W_s * x) & \text{if there is a change in dimensionality} \end{cases} \quad (8)$$

where y is the output of the residual block, x is the input to the residual block, $F(x, \{W_i\})$ represents the transformations in the main branch, consisting of convolutional layers followed by batch normalization and ReLU activation, W_s are the filters of the 1×1 convolution used in the shortcut connection when dimensionality adjustment is required, $*$ denotes the convolution operation, and ReLU is the activation function applied after the addition.

This formulation ensures that, regardless of changes in dimensionality, the architecture maintains an effective shortcut connection that promotes gradient flow and stabilizes network training.

- Identity Blocks

Identity blocks are a subset of residual blocks used when the dimensionality of the input and output of the block does not change. In this scenario, the shortcut connection is simply the identity, meaning that the input is added directly to the output of the main branch without any additional transformation.

The operation for the identity block is as follows:

$$y = ReLU(F(x, \{W_i\}) + x) \quad (9)$$

where y represents the final output of the identity block, x is the input to the identity block, $(x, \{W_i\})$ denotes the transformations in the main branch, consisting of convolutional layers followed by batch normalization and ReLU activation, and ReLU is the activation function applied after the addition.

- Projection Blocks

When the stride exceeds 1, a projection block adjusts the input dimensions to match the main branch's output dimensions. This is essential to ensure dimensional compatibility during the addition of the branches. The operation in a projection block is defined as follows:

$$y = ReLU(F(x, \{W_i\}) + Conv_{1x1}(x, stride)) \quad (10)$$

where y represents the output of the projection blocks, x is the input to the residual block, $F(x, \{W_i\})$ represents the transformations in the main branch, consisting of 3×3 convolutional layers followed by batch normalization and ReLU activation, $Conv_{1\times1}(x, stride)$ is a 1×1 convolution operation with a stride more significant than one used to adjust the dimensions of x and $ReLU$ is the activation function applied after the addition.

This block allows for spatial resolution adjustments and/or an increase in the number of filters, modifying the dimensions of the shortcut connection to maintain dimensional consistency.

- Global Pooling Layer

After applying several residual blocks, a Global Average Pooling (GAP) layer reduces the spatial dimensions of the output from the last residual block. This operation calculates the average of the activations for each channel, producing a feature vector of length C,

where C is the number of output channels. Mathematically, the GAP operation is defined as follows:

$$GAP(x) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x[i, j, c] \tag{11}$$

where H and W are the spatial dimensions of the output from the residual block, and $x[i, j, c]$ is the activation value at position (i, j) in channel c.

The result is a feature vector of dimension c, where each element represents the average activation of a specific channel. This layer transforms spatial feature maps into global feature vectors, facilitating the connection with the dense output layer for classification.

Pooling layers have gained popularity in modern convolutional neural network architectures for reducing data dimensionality and decreasing the possibility of overfitting. Global average pooling is commonly used after the last convolutional layer to flatten the feature maps instead of aggregating many dense layers that significantly increase the model's parameters. By averaging spatially, the network extracts global information from each channel without assigning weight to specific locations, converting the feature map into a vector with a size equal to the number of channels, which is very useful for classification tasks. Although max pooling effectively highlights more substantial features considering specific regions of the input data, its use has been mainly for intermediate convolutional layers to reduce the spatial size of feature maps (width and height), retaining crucial local information that will be entered into the next convolutional layer.

- Output Layer

The output of the Global Average Pooling layer is connected to a fully connected layer with a single neuron and sigmoid activation, which produces the probability for binary classification. The operation in the output dense layer is defined as follows:

$$y = \sigma(W \cdot x + b) \tag{12}$$

where y is the output of the dense layer, x is the feature vector resulting from the GAP layer, W is the weight vector of the dense layer, b is the bias, and σ is the sigmoid function, defined as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{13}$$

The sigmoid function maps any real value to a range between 0 and 1, which is interpreted as the probability of belonging to one of the classes (organic vs. inorganic). By using a single neuron in the output layer with a sigmoid activation function, the model reduces the number of parameters needed, resulting in potentially faster training with less risk of overfitting. In addition, a single output value directly indicates the probability p of the positive class, and its complement $1 - p$ corresponds to the negative class, so a second neuron is not required for this case.

- Loss Function and Optimization

Binary Focal Crossentropy is used as the loss function to train the network, which is especially effective in problems with imbalanced classes. This choice is based on the ability of Binary Focal Crossentropy to handle class imbalance by focusing on the more complex examples and reducing the influence of well-classified examples. This loss function is defined as follows:

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} log(p_t) \tag{14}$$

where α is an adjustment factor to compensate for the class imbalance, p_t is the predicted probability for the true class and γ is the modulation factor that reduces the contribution of well-classified examples and focuses learning on complex examples.

These hyperparameters have been selected based on previous studies demonstrating their effectiveness in imbalanced binary classification tasks [26]. The combination of $\alpha = 0.25$ and $\gamma = 2.0$ allows the model to improve its performance in detecting waste from the minority class, preventing it from being biased toward the majority class.

The model is optimized using the Stochastic Gradient Descent (SGD) with a momentum of 0.9 and an adjustable learning rate. The weight update is performed according to the following formula:

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla L(\theta) \tag{15}$$

$$\theta_t = \theta_{t-1} - \eta v_t \tag{16}$$

where v_t is the velocity at time, $\beta = 0.9$ is the momentum factor, which helps accelerate convergence in relevant directions and smooth out oscillations during training, $\nabla L(\theta)$ is the gradient of the loss function concerning the weights θ and η is the learning rate that determines the size of the update steps.

The SGD optimizer's hyperparameters were selected and adjusted based on widely accepted recommendations from the literature on deep neural network training. The momentum was set to 0.9 since this parameter facilitates faster convergence and smooths out possible oscillations by accumulating previous information about the direction of the gradient. The Nesterov momentum variant was activated, allowing more accurate anticipation of gradient directions. The initial learning rate was set to 0.01, providing a gradual and stable step size reduction as the training progressed. Finally, to improve the stability and efficiency of the learning process further, ReduceLROnPlateau (https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau (accessed on 8 March 2025)) was implemented, which dynamically adjusts the learning rate if the validation loss does not improve after several consecutive epochs. This combination of strategies allows the model to converge efficiently towards a global optimum, thus avoiding overfitting problems and improving the model's accuracy in binary waste classification.

2.4. Evaluation of the Neural Network

The performance evaluation was based on accuracy, the confusion matrix, recall, F1-score, precision, and cross-entropy loss observed during the model's training, validation, and testing.

Accuracy represents the proportion of correct classifications made by the waste classification model, and it can be defined using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{17}$$

where True Positive (*TP*) refers to the number of images with organic elements that were correctly predicted, True Negative (*TN*) indicates the number of inorganic waste images that were correctly predicted, False Positive (*FP*) indicates the number of inorganic waste images that were predicted as belonging to the organic class, and False Negative (*FN*) indicates the number of organic class images that were predicted as inorganic.

Precision represents the proportion of genuinely positive samples among all samples predicted as positive, and it can be formulated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

The recall represents the proportion of all positive samples that are correctly predicted as positive, and it can be expressed as follows:

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

The F1-score is the harmonic mean of precision and recall, where an F1-score reaches its best value at 1 and its worst value at 0.

$$F1\text{-}score = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{20}$$

3. Results

For the development of the R3sNet neural network model, Python version 3.10.14 was chosen for its versatility and widespread adoption in machine learning applications and data analysis. The implemented libraries and environments included TensorFlow 2.10.1, Keras 2.10.0, NumPy 1.26.4, Pandas 1.4.4, Matplotlib 3.9.0, and Seaborn 0.13.2.

The model was developed and executed in a Jupyter Notebook-based programming environment v1.96. The system was implemented and trained on a machine equipped with an Intel Core i7 processor, an NVIDIA GTX 1660Ti GPU with 6 GB of dedicated memory, 23 GB of RAM, and 1 TB of SSD storage.

Subsequently, the initial training dataset was divided into two subsets: training and validation. For this purpose, the train_test_split function from the scikit-learn library was used, applying a 15% data partition for validation and ensuring an equitable distribution of classes using the stratify parameter. The split was performed with a fixed seed (random_state = 42) to guarantee the reproducibility of the results.

Performance of the R3sNet and ResNet50 Models

ResNet18 was chosen as the base architecture because it is the smallest version of ResNet reported in [23]. However, we could not test with ResNet18 due to limitations in accessing a pretrained implementation compatible with the Tensorflow framework. Although ResNet-50 is larger than our model, it is a widely used state-of-the-art network, and its greater representational capacity allows us to evaluate how a deeper model with more parameters performs when classifying organic and inorganic waste compared to our lighter model. This provides insight into the trade-off between complexity and performance, which is relevant for practical applications where computational resources may vary. ResNet-50 has a greater capacity to learn complex features and generalize across diverse datasets. Comparing our model with ResNet-50 allows us to analyze whether our lighter design, which is trained end-to-end (no pre-trained weights are required), can achieve a similar level of generalization, especially on a challenging dataset such as images of organic and inorganic waste, where visual characteristics (color, texture, shape) vary widely.

Ten training repetitions were conducted for the R3sNet and pre-trained ResNet50 models using the same parameters, as shown in Table 4, and dataset to ensure a fair comparison. This resulted in a reduction in hyperparameters by 98.60% and 65.17% in GFLOPS. The evaluated performance metrics included precision, recall, F1-score, and accuracy. Additionally, loss and accuracy curves were recorded, with the model converging at epoch six, where the loss and accuracy metrics stabilized. Although minor fluctuations occurred afterward, these were insignificant, suggesting that the model had reached a point of stability; therefore, training was stopped after 30 epochs. Table 5 displays the results obtained on the test set, with the calculated mean and standard deviation.

Table 4. Hyperparameters used in R3sNet and pre-trained ResNet50.

Hyperparameter	Value
Input	224, 224, 3
Batch	16
Activation	Sigmoid
Optimizer	SGD
learning_rate	1×10^{-3}
Epochs	30

Table 5. Performance metrics for R3sNet and ResNet50.

Dataset	Metric	R3sNet	Pretrained ResNet50
Train	Accuracy	0.8941 ± 0.0053	0.8026 ± 0.0109
	Loss	0.0701 ± 0.0027	0.4281 ± 0.0170
Validation	Accuracy	0.8901 ± 0.0156	0.8128 ± 0.0162
	Loss	0.0758 ± 0.0102	0.4267 ± 0.0211
Test	Accuracy	0.8633 ± 0.156	0.8167 ± 0.0115
	Loss	0.0965 ± 0.0109	0.4011 ± 0.0173

The results show that R3sNet consistently outperforms ResNet50 across the three evaluated metrics. R3sNet achieves an average training accuracy of 0.89, with a standard deviation of 0.005, compared to 0.83 for ResNet50. Similarly, R3sNet presents average values of 0.89 and 0.86 in validation and testing, respectively, while ResNet50 obtains 0.81 in both cases.

Furthermore, the increase in accuracy and the minor standard deviations indicate that the proposed model's performance is stable over iterations (for example, 0.0156 versus 0.016–0.011 in validation and testing), suggesting that R3sNet achieves better results and does so consistently.

Overall, these findings indicate that the lightweight architecture of R3sNet offers an improvement over standard ResNet50, at least within the dataset and experimental conditions used.

Figures 4 and 5 show the graphs of the mean and standard deviation obtained during training, validation, and testing for R3sNet and ResNet50, where the superior performance of the proposed model can be observed.

The R3sNet model during training shows a mean accuracy of 0.84 in the initial epoch. This relatively high starting value suggests that the model can capture significant patterns from the first training steps. During the first five epochs, a rapid increase in accuracy is observed, reaching 0.89. Subsequently, the growth slows, stabilizing at values close to 0.90 in the final epochs. This behavior is characteristic of a model that has captured the most relevant patterns in the training set and is refining its classification ability in the final stages. The low standard deviations associated with this metric, ranging between 0.005 and 0.007, indicate that the results obtained are highly consistent across the 10 runs performed, reinforcing the model's robustness on the training set.

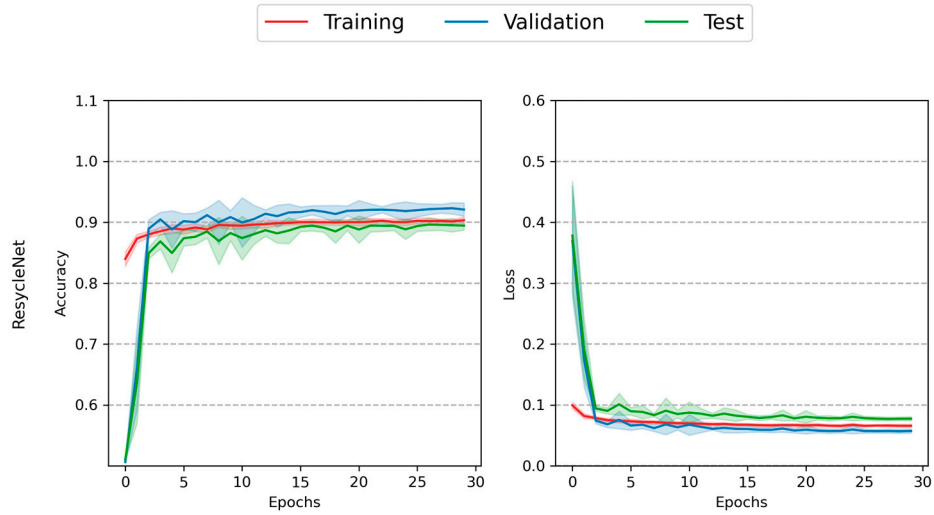

Figure 4. Graph displaying the mean and standard deviation of R3sNet across ten runs.

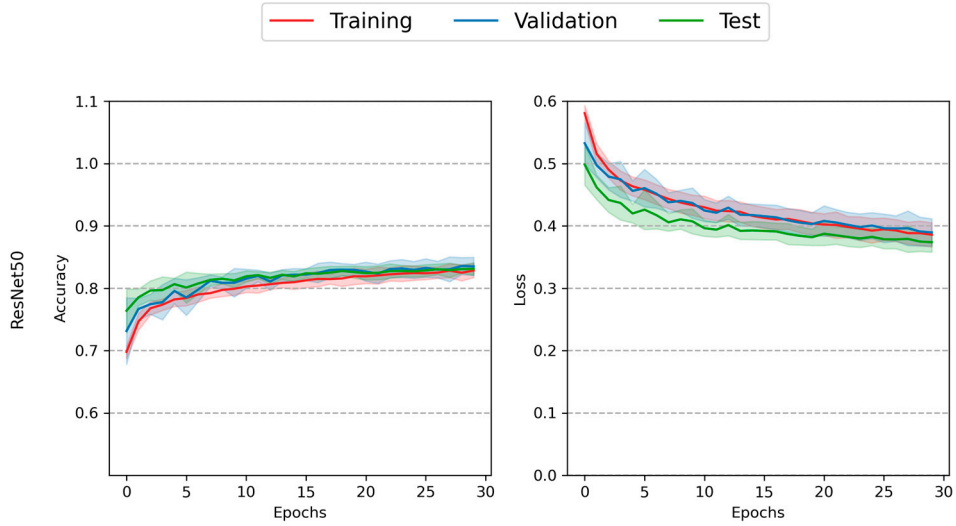

Figure 5. Graph of the mean and standard deviation using pretrained ResNet50 across ten runs.

Regarding the validation set, a similar behavior is observed to that of the training set, but with a lower starting point (0.50 at epoch 0). A notable increase is observed in the initial epochs (1 to 3), reaching values of 0.88–0.90. From epoch five onward, the validation accuracy stabilizes between 0.90 and 0.92, remaining constant for the rest of the training. This behavior suggests that the model is learning in a generalized manner rather than simply memorizing the training set. The standard deviations in validation (0.01–0.02) are slightly higher than in training during the early epochs, but they progressively decrease as the model converges, indicating consistent results. Its accuracy on the test set starts at 0.51 in epoch zero and shows a sustained increase during the first five epochs, reaching 0.85–0.87. In the final epochs, this metric stabilizes between 0.88 and 0.89, which is very close to the validation results (0.91). The 2% difference between validation and test is reasonable, reflecting adequate model generalization, suggesting that R3sNet can effectively apply the patterns learned from the training data to classify previously unseen examples.

ResNet50 starts with a mean accuracy of 0.70 at epoch 0. This value reflects a moderate starting point, consistent with this model's greater depth and complexity, which requires more time to adjust its initial parameters. As training progresses, a gradual and steady increase is observed during the first 10–15 epochs, reaching values close to 0.81, indicating that the model has learned most of the relevant patterns in the dataset. The low standard

deviations (between 0.01 and 0.02) indicate performance stability across different runs, suggesting that the results are reproducible and consistent. Its validation accuracy shows behavior like training, although it starts from a slightly higher value (0.73 at epoch 0). The model experiences a moderate increase in the early epochs, reaching 0.77–0.79 between epochs 5 and 10. Subsequently, this metric converges to values of 0.82–0.83 in the later iterations, reflecting a good ability to generalize the patterns learned during training. The closeness between validation and training accuracy suggests that the model does not overfit, even after 30 epochs.

Regarding consistency, the standard deviations in validation decrease from 0.02 to 0.03 during the early epochs and to 0.01 toward the end, demonstrating that it becomes more predictable as it converges. The test accuracy begins at 0.76 and improves gradually in the early epochs, like validation. In the final iterations, this metric reaches values closely aligned to validation values. The difference between the two metrics is less than 1%, confirming that the model generalizes effectively and consistently.

The minimum observed loss reaches 0.4, indicating that ResNet50 still presents a significant margin of error. This loss value suggests that the model lacks complete confidence in its predictions, as there is a considerable probability of incorrect classification. Compared to R3sNet, which has a loss of less than 0.1, R3sNet performs better in accuracy and reliability by leveraging image features more effectively and achieving greater generalization, resulting in a reduced loss and higher confidence in the predictions.

In this study, 10 repetitions were carried out, and to ensure the robustness and reliability of the results, the images in the training and test sets were selected randomly. This approach allowed the model's performance to be evaluated under different data distributions and avoided possible biases associated with a specific partition. The best case among the 10 repetitions was selected for the results, corresponding to the instance that maintained the lowest loss value (across the training, validation, and test sets). This approach ensures that the reported results reflect the maximum performance achieved by R3sNet. Additionally, the confusion matrices presented in Figure 6 correspond to this best case for R3sNet and pretrained ResNet50, providing a clear and representative view of the model's ability to classify the images correctly.

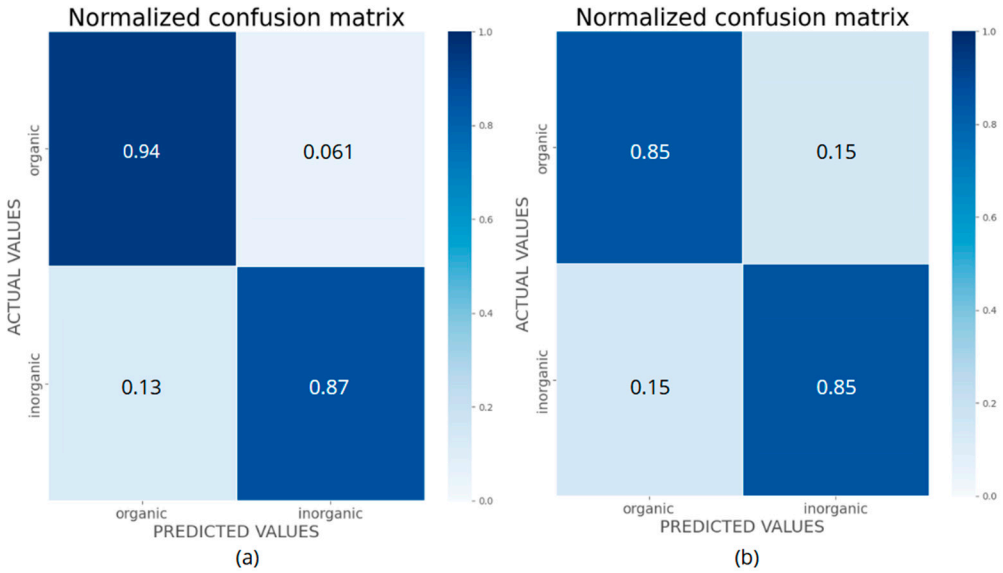

Figure 6. Confusion matrix of the models (**a**) R3sNet and (**b**) pretrained ResNet50.

4. Discussion

The classification of MSW has evolved significantly due to advances in convolutional neural networks (CNNs). This study implemented an optimized version of ResNet called R3sNet, specifically designed to address binary waste classification with computational efficiency and high accuracy. Several relevant aspects stand out when comparing the results obtained with related research. In terms of performance, R3sNet achieved an accuracy of 87% for organic waste classification and 94% for inorganic waste, surpassing the results obtained by conventional CNN models in [23], which averaged 80.88% accuracy in both classifications using a lightweight five-layer architecture with lower resolution images. On the other hand, the authors of [14] implemented a small CNN that achieved 96.77% on their custom dataset and 93.72% on the TrashNet set; they employed advanced preprocessing techniques such as adaptive brightness algorithms and Canny operators for background cropping, which may have contributed to improved accuracy. In comparison, R3sNet, by integrating residual blocks with shortcut connections, ReLU activation functions, and batch normalization, proves to be an efficient solution that mitigates common issues such as vanishing gradients and overfitting. Moreover, the Binary Focal Crossentropy loss function effectively handled class imbalance.

In terms of computational efficiency, R3sNet seeks to balance performance. In contrast to models like DenseNet121 trained for MSW [5,6] with many parameters (over eight million), R3sNet significantly reduces computational requirements while maintaining competitive accuracy. This is especially useful for implementations in resource-limited environments like IoT devices. Ref. [23] used a dataset taken from [18] that contained 25,077 images, which was one of the datasets also used in this study; however, duplicate images were identified and removed during the dataset curation phase. This step is crucial to avoid training biases and ensure the model performance calculation is representative and reliable. Furthermore, a consistent evaluation using a specific data combination allowed for more robust and generalizable results.

Although the volume of data used is relatively limited, it was considered sufficient to effectively train the R3sNet model because of the strategies implemented to maximize sample diversity. Online data augmentation techniques and duplicate removal enhanced the variability of the training set without necessitating additional data collection. These strategies increased the compelling diversity of examples submitted to the model, mitigating the limitations arising from fewer samples and reducing the risk of overfitting. Online data augmentation applies new transformations to the original images, generating variations at each training epoch and ensuring different images at each iteration. Thus, R3sNet captured a more robust set of overall features. A more extensive dataset could further improve the performance and generalizability of the model, as a more significant number of examples tends to cover the variability of the residuals in a more complex manner. However, the results indicate that the model achieved competitive performance despite the limited size. This suggests that the amount of available data enriched by data augmentation and duplicate cleaning was adequate to train R3sNet effectively, achieving a satisfactory balance between model complexity and the information provided.

Traditional convolutional architectures with many parameters often overfit when access to limited datasets is available. Public datasets for MSW classification are frequently relatively small; for instance, around 25,000 images are usually available for binary tasks like differentiating between organic and inorganic. This stands in stark contrast to extensive datasets like ImageNet. Training networks with millions of parameters from scratch in these situations can result in models memorizing the training data rather than learning generalizable patterns. While conventional models such as VGG-16, ResNet50, or Dense-Net-121 have exhibited high accuracy in other tasks, they comprise many layers

and parameters (with VGG-16 containing approximately 138 million parameters) and require substantial computing power. Consequently, they are unsuitable for applications in resource-constrained environments such as embedded devices. Even attempts to reduce or freeze layers in pre-trained models often yield bulky models that adversely affect storage and operational efficiency.

One limitation of the current study is that the model was tested on only two classes (organic and inorganic), which might not entirely reflect its potential in a multi-class scenario. Therefore, future work is planned to extend the analysis to a multiclass problem, which may require adjustments in the model architecture and hyperparameter tuning to optimize its performance in more complex tasks.

R3sNet exhibits exceptional accuracy, featuring swift initial learning, stabilization during the final epochs, and strong generalization capabilities. The optimized architecture and appropriate training configuration enhance these results, making it a practical choice for MSW classification tasks.

The results demonstrate R3sNet's potential for efficient and accurate MSW classification. However, future research could focus on validating the model with more extensive and diverse datasets and incorporating advanced preprocessing techniques to improve its performance, e.g., analyzing residues in images with complex backgrounds and variable lighting conditions by applying transfer learning using our pre-trained network.

R3sNet has excellent potential for real-world waste sorting. Its lightweight design allows it to be integrated into automatic sorting systems (conveyor belts or recycling machines), reducing costs and improving waste separation efficiency. This could reduce the amount of waste in landfills and support circular economy objectives, especially in regions with limited resources. However, challenges such as upfront hardware costs (cameras and other sensors) and environmental variations could affect its performance, requiring optimizations such as data collection under diverse conditions.

Compared to recent studies, R3sNet aligns with the trend toward more efficient models. For example, the work by [27] used MobileNet with transfer learning and achieved 93.35% accuracy on 2400 images. Our model shows lower accuracy, possibly due to the lack of transfer learning. Ref. [27] highlights the effectiveness of reusing pre-trained weights, suggesting that R3sNet could benefit from this technique to improve its performance, especially with small datasets.

Our model performance is slightly lower than that of [28], who achieved an accuracy of 93.28% in classifying organic and recyclable waste using an improved convolutional neural network architecture. This difference could be due to the different optimization and regularization strategies employed. While [28] used deep convolutional layers, LeakyReLU, and dropout, our approach does not incorporate these improvements, which limits feature extraction. Furthermore, the greater quantity and quality of data used by [28] (25,077 images, 70% training, 30% test) suggests that a larger or more balanced dataset could improve the performance of our architecture. Incorporating modifications such as LeakyReLU or adjusted dropout rates could benefit our proposal, especially in scenarios with high variability of waste. Although our accuracy is slightly lower, R3sNet is optimized for efficiency with significantly fewer parameters, making it suitable for environments with limited computational resources. In conclusion, R3sNet offers an efficient and scalable solution for MSW binary classification, outperforming heavier models such as ResNet50 and being competitive against recent approaches; while the improved DCNN of [28] prioritizes accuracy in more demanding contexts for real-world applications, R3sNet is ideal in low-cost environments and DCNN could be preferred where error reduction is a priority. Future research could combine the efficiency of R3sNet with the accuracy of models such as DCNN, optimizing its implementation in waste management systems under variable conditions.

5. Conclusions

This study presents an optimized architecture based on ResNet, called R3sNet, which is explicitly designed for binary MSW classification. The results highlight the proposed model's high performance, achieving an accuracy of 87% in organic waste classification and 94% in inorganic waste. These metrics surpass those of ResNet50 and demonstrate that a relatively small architecture featuring residual blocks, ReLU activation functions, batch normalization, and the Binary Focal Crossentropy loss function significantly contributes to the model's performance and stability.

The R3sNet architecture offers competitive accuracy and optimizes the use of computational resources, making it suitable for deployment in hardware-constrained environments, such as IoT devices. Additionally, its design facilitates generalization and minimizes common issues in deep networks, such as overfitting and vanishing gradients. Compared to ResNet50, R3sNet reduces hyperparameters by 98.60% and GFLOPS by 65.17%.

It is important to emphasize that the efficient design of R3sNet contributes to sustainability in several ways. First, it reduces the burden on landfills, promotes a circular economy, and enables more effective waste sorting. Second, the reduced computational demand is reflected in energy consumption, making it suitable for implementation in environments with hardware limitations, such as IoT devices used in smart city applications. These contributions indicate that the research contributes to SDGs 11, 12, and 13, reinforcing the model's potential for sustainable urban development.

In conclusion, R3sNet is a significant advance in the intelligent management of MSW. It offers a solution to the challenges that arise, ensuring that technological development goes hand in hand with sustainable development.

Author Contributions: Conceptualization, M.C.-B., J.F.-P. and V.M.R.-J.; methodology, J.F.-P., M.C.-B. and V.M.R.-J.; software, D.E.G.-V. and S.R.Z.-B.; validation, M.C.-B., J.F.-P., V.M.R.-J. and C.M.-D.; formal analysis, J.F.-P., C.M.-M. and C.M.-D.; investigation, V.M.R.-J. and J.F.-P.; resources, V.M.R.-J., J.F.-P. and C.M.-M.; data curation, V.M.R.-J. and J.F.-P.; writing—original draft preparation, V.M.R.-J., M.C.-B. and J.F.-P.; writing—review and editing, J.F.-P., V.M.R.-J. and M.C.-B.; visualization, C.M.-D. and C.V.M.-V.; supervision, C.V.M.-V.; project administration, M.C.-B., J.F.-P. and V.M.R.-J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset can be obtained from the following link: https://github.com/victorMRJDev/datasetR3sNet, accessed on 9 February 2025.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kaza, S.; Yao, L.C.; Bhada-Tata, P.; Van Woerden, F. *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*; World Bank: Washington, DC, USA, 2018. [CrossRef]
2. Lin, K.; Zhao, Y.; Kuo, J.H.; Deng, H.; Cui, F.; Zhang, Z.; Zhang, M.; Zhao, C.; Gao, X.; Zhou, T.; et al. Toward Smarter Management and Recovery of Municipal Solid Waste: A Critical Review on Deep Learning Approaches. *J. Clean. Prod.* **2022**, *346*, 130943. [CrossRef]
3. Zhang, Q.; Zhang, X.; Mu, X.; Wang, Z.; Tian, R.; Wang, X.; Liu, X. Recyclable Waste Image Recognition Based on Deep Learning. *Resour. Conserv. Recycl.* **2021**, *171*, 105636. [CrossRef]
4. Liu, X.; Zhou, Q.; Zhao, J.; Shen, H.; Xiong, X. Fault Diagnosis of Rotating Machinery under Noisy Environment Conditions Based on a 1-D Convolutional Autoencoder and 1-D Convolutional Neural Network. *Sensors* **2019**, *19*, 972. [CrossRef] [PubMed]
5. Mao, W.L.; Chen, W.C.; Wang, C.T.; Lin, Y.H. Recycling Waste Classification Using Optimized Convolutional Neural Network. *Resour. Conserv. Recycl.* **2021**, *164*, 105132. [CrossRef]

6. Zhang, Q.; Yang, Q.; Zhang, X.; Bao, Q.; Su, J.; Liu, X. Waste Image Classification Based on Transfer Learning and Convolutional Neural Network. *Waste Manag.* **2021**, *135*, 150–157. [CrossRef] [PubMed]
7. Malik, M.; Sharma, S.; Uddin, M.; Chen, C.-L.; Wu, C.-M.; Soni, P.; Chaudhary, S. Waste Classification for Sustainable Development Using Image Recognition with Deep Learning Neural Network Models. *Sustainability* **2022**, *14*, 7222. [CrossRef]
8. Shi, C.; Tan, C.; Wang, T.; Wang, L. A Waste Classification Method Based on a Multilayer Hybrid Convolution Neural Network. *Appl. Sci.* **2021**, *11*, 8572. [CrossRef]
9. Bin Islam, S.; Sumon, S.I.; Majid, M.E.; Kashem, S.B.A.; Nashbat, M.; Ashraf, A.; Khandakar, A.; Kunju, A.K.A.; Hasan-Zia, M.; Chowdhury, M.E. ECCDN-Net: A Deep Learning-Based Technique for Efficient Organic and Recyclable Waste Classification. *Waste Manag.* **2025**, *193*, 363–375. [CrossRef] [PubMed]
10. Keskin, E.F.; İşleyen, O.; Demirhan, H.; Keskin, Ş.R. Trash Classification Using Deep Learning Models. In Proceedings of the 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), Las Vegas, NV, USA, 24–27 July 2023; pp. 318–323.
11. Abdu, H.; Noor, M.H.M. Domestic Trash Classification with Transfer Learning Using VGG16. In Proceedings of the 2022 IEEE 12th International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 21–22 October 2022; pp. 137–141.
12. Azis, F.A.; Suhaimi, H.; Abas, E. Waste Classification Using Convolutional Neural Network. In Proceedings of the 2020 2nd International Conference on Information Technology and Computer Communications, Kuala Lumpur, Malaysia, 12–14 August 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 9–13.
13. Lin, K.; Zhao, Y.; Gao, X.; Zhang, M.; Zhao, C.; Peng, L.; Zhang, Q.; Zhou, T. Applying a Deep Residual Network Coupling with Transfer Learning for Recyclable Waste Sorting. *Environ. Sci. Pollut. Res.* **2022**, *29*, 91081–91095. [CrossRef] [PubMed]
14. Yang, Z.; Xia, Z.; Yang, G.; Lv, Y. A Garbage Classification Method Based on a Small Convolution Neural Network. *Sustainability* **2022**, *14*, 14735. [CrossRef]
15. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
16. Thung, G. Garythung/Trashnet: Dataset of Images of Trash; Torch-Based CNN for Garbage Image Classification. Available online: https://github.com/garythung/trashnet?tab=readme-ov-file (accessed on 20 December 2024).
17. Xiaohoua. Garbage_HuaWei. Available online: https://www.kaggle.com/datasets/xiaohoua/garbage-huawei (accessed on 9 December 2024).
18. Sekar, S. Waste Classification Data. Available online: https://www.kaggle.com/datasets/techsash/waste-classification-data (accessed on 20 December 2024).
19. Kunwar, S. Garbage Dataset. Available online: https://www.kaggle.com/dsv/10182596 (accessed on 9 December 2024).
20. Mohamed, M. Garbage Classification (12 Classes). Available online: https://www.kaggle.com/datasets/mostafaabla/garbage-classification (accessed on 20 December 2024).
21. Garbage Pictures for Classification. Available online: https://www.kaggle.com/datasets/mascot9183/garbage-pictures-for-classification?select=Garbage+classification (accessed on 9 December 2024).
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
23. Nnamoko, N.; Barrowclough, J.; Procter, J. Solid Waste Image Classification Using Deep Convolutional Neural Network. *Infrastructures* **2022**, *7*, 47. [CrossRef]
24. Xie, S.; Girshick, R.; Dollar, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
25. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
26. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
27. Suganda Girsang, A.; Pratama, H.; Santo Agustinus, L.P. Classification Organic and Inorganic Waste with Convolutional Neural Network Using Deep Learning. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 343–348.
28. Chhabra, M.; Sharan, B.; Elbarachi, M.; Kumar, M. Intelligent Waste Classification Approach Based on Improved Multi-Layered Convolutional Neural Network. *Multimed. Tools Appl.* **2024**, *83*, 84095–84120. [CrossRef]