Received November 15th 2025; accepted December 24th 2025. Date of publication December 31st 2025
Digital Object Identifier: https://doi.org/10/25047/jtit.v12i2.458

CNN Implementation in Progressive Web App for Automatic Garbage Classification using TensorFlow.js

EKA SETYABUDI[1], NOORA QOTRUN NADA[2], MEGA NOVITA[3]

[1]Universitas PGRI Semarang, Semarang, Indonesia
[2]Universitas PGRI Semarang, Semarang, Indonesia
[3]Universitas PGRI Semarang, Semarang, Indonesia

CORESPONDING AUTHOR: EKA SETYABUDI (email:22670144@upgris.ac.id)

ABSTRACT The substantial and continuously increasing volume of global solid waste has evolved into a critical environmental challenge. This issue is further exacerbated by the inherent inefficiency of conventional manual sorting techniques, which are not only costly but also pose significant health risks to workers due to exposure to hazardous materials. To address the limitations of existing server-dependent systems, this research develops and evaluates an automated, client-side waste classification system. We employ a Convolutional Neural Network (CNN) based on the VGG16 architecture, integrated into a Progressive Web App (PWA) to ensure broad accessibility across devices without requiring native installation. The methodology involves retraining the VGG16 model using transfer learning on a validated public dataset of 10,365 images, categorized into organic and inorganic waste. Preprocessing steps included resizing images to 224x224 pixels, pixel normalization, and data augmentation to enhance model robustness against real-world variations. Subsequently, the trained model was converted into the web-compatible TensorFlow.js format and deployed within a PWA framework that utilizes Service Workers and IndexedDB caching mechanisms to enable offline functionality. Results indicate that despite the computational challenges posed by the model's large size, the system successfully performed efficient client-side inference by prioritizing WebGL backend for GPU acceleration. The model achieved an overall accuracy of 94% on the test dataset, with a precision of 95% for inorganic waste. These findings confirm that deploying high-accuracy CNN models at the edge using PWA and TensorFlow.js is a feasible and promising strategy for practical, technology-based waste management and environmental education.

KEYWORDS: Waste Classification; Convolutional Neural Network; VGG16; Progressive Web App; TensorFlow.js.

I.INTRODUCTION

The simultaneous impact of population growth and urbanization on a global scale has increased the generation of solid waste [1], [2]. Ineffective waste management, particularly in sorting and recycling, poses a global environmental and social challenge [1], [3], [4]. This contributes to increased greenhouse gas emissions, environmental pollution, and the depletion of natural resources [5].

The manual process of waste sorting and collection has many shortcomings [4], [6], [7]. In addition to high costs, this method poses health problems for workers due to exposure to pathogens and toxic substances [1], [5]. This problem is exacerbated by low public awareness of separating household waste [8], which contaminates recyclable materials and increases the volume of waste sent to landfills.

In this context, there is a need for more sophisticated and efficient automated systems with a broader user interface [2], [3]. To achieve this, the system must be intelligent, easy to operate, lightweight, and able to run on a variety of devices. This solution can be accessed using a browser-based cross-platform PWA without requiring additional app installation. Furthermore, offline features and home screen deployment provide a native app-like experience.

Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have made significant contributions to image classification [20], including trash classification [8], [9], [10], [11], [12], [13]. Implemented CNN architectures such as ResNet, Inception, MobileNet, VGG, and TrashNet have achieved excellent accuracy [8], [9], [12], [14]. However, most classification systems still rely on servers or native applications... [lanjutkan sampai] ...In general, the biggest challenge is how to combine large models with the accuracy, efficiency, and lightweight features of a PWA platform.

Recent advances in automatic waste classification have demonstrated promising results using various CNN architectures. For instance, Nahiduzzaman et al. [3] developed a server-based system achieving high accuracy but requiring continuous internet connectivity. Similarly, White et al. [6] proposed WasteNet for edge deployment using IoT hardware, while Yong et al. [8] utilized MobileNetV2 prioritizing lightweight deployment on mobile devices at the expense of accuracy in complex environments. However, these approaches either depend on specialized hardware [6], [19] or sacrifice classification performance for efficiency [8]. Furthermore, existing browser-based solutions remain limited due to model size constraints and lack of robust offline capabilities [16].

The gap addressed by this research lies in the absence of a truly accessible, client-side waste classification system that maintains high accuracy without requiring native app installation or continuous server connectivity. While previous studies have achieved strong performance through server-side processing [3] or dedicated edge devices [6], there is limited research on deploying large, high-accuracy CNN models directly in Progressive Web Apps with full offline functionality.

The novelty of this work is threefold: (1) it successfully integrates the computationally intensive VGG16 architecture into a PWA environment using TensorFlow.js with GPU acceleration, (2) it implements comprehensive offline capabilities through Service Worker caching and IndexedDB, enabling waste classification without internet dependency, and (3) it provides a replicable framework for deploying large deep learning models at the edge through web browsers, making the technology accessible across devices without installation barriers. This approach specifically addresses the limitation noted by Nahiduzzaman et al. [3] regarding server dependency and extends beyond the lightweight-focused approaches [8] by demonstrating that high-accuracy models can be practically deployed client-side with appropriate optimization strategies.

The selection of VGG16 over lighter alternatives like MobileNet is justified by its superior feature extraction capabilities for diverse waste images, as demonstrated in recent comparative studies [11], [12]. Despite its computational demands, VGG16's deeper architecture with 13 convolutional layers provides robust pattern recognition essential for handling 'real-world' waste images with varying lighting conditions, angles, and backgrounds [16]. The choice of PWA as the deployment platform addresses the digital divide by eliminating app store barriers and installation friction, while enabling cross-platform compatibility and offline functionality through Service Workers [17], critical features for reaching diverse user demographics in waste management education and practice.

This research aims to bridge this gap by integrating the VGG16 model into a PWA. This approach emphasizes not only classification accuracy but also model loading efficiency, offline capabilities, and the overall user experience. Thus, this solution is expected to be a practical and inclusive alternative for technology-based waste management.

II.METHOD

2.1. Data Collection and Preprocessing

This study used a public waste classification dataset sourced from Kaggle (https://www.kaggle.com/code/divaanggreinihar ahap/klasifikasi-gambar/input). The dataset was then uploaded to Google Drive for processing. Of the total available dataset, this study used 10,365 preprocessed and validated images, divided into two classes: 5,709 images of inorganic waste and 4,656 images of organic waste. This dataset selection prioritized those with high image diversity, including "real-world" conditions, to increase model robustness [2], [4], [16].

It should be noted that many waste classification datasets face the problem of class imbalance [3], where some types of waste (e.g., plastic and paper) dominate while other categories (e.g., hazardous materials) are underrepresented. Without intervention, this imbalance is likely to cause AI models to develop algorithmic bias [3], underperforming minority classes despite demonstrating high overall accuracy. Therefore, the need for datasets that support a more balanced class distribution, or the careful application of data augmentation (such as oversampling for minority classes) [5], [15],[19], is crucial to ensure fairness and efficiency across all bin categories. Using datasets with "real-world" images is also crucial to improve the model's generalizability to real-world conditions [16].

The data preprocessing steps include:

1. Dataset Preparation: Each dataset is automatically divided into training, validation, and testing parts in an 80:10:10 ratio.
2. Image Scaling: All images are uniformly resized to 224x224 pixels with 3 color channels (RGB), in accordance with the standard input specifications for the VGG16 architecture.
3. Pixel Normalization: Normalize pixel intensity values that were previously within a normalized range by dividing each value by 255.
4. Zero-Centering and Color Sequence Conversion: Calculate the RGB average of the

dataset and subtract pixels from that average value (zero-centering). If the model expects a BGR channel sequence, the image will be converted from RGB to BGR.
5. Data Augmentation: Apply rotation, width/height shift, zoom, and horizontal flip to reduce overfitting and increase dataset variation.

Before inference, the input image from the camera is converted into a tensor, resized to 224x224 pixels using bilinear interpolation, and normalized to a range of [0, 1]. The implementation uses tf.tidy to prevent memory leaks during this process, as demonstrated in Listing 1.

Listing 1. Client-Side Image Preprocessing

```
const preprocessImage = (imageElement) => {
        return tf.tidy(() => {
 // Convert DOM element to Tensor
    const img = tf.browser.fromPixels(imageElement);

 // Resize to VGG16 input standard (224x224)
    const resized = tf.image.resizeBilinear(img, [224, 224]);

 // Normalize pixel values (0-255 -> 0-1)
// and add batch dimension [1, 224, 224, 3]
    const normalized = resized.toFloat().div(255.0).expandDims(0);

        return normalized;
      });
    };
```

2.2. Model Architecture and Training

The VGG16 Convolutional Neural Network model was used as the feature extractor [7], [11]. This architecture consists of 13 convolutional layers composed of 3x3 filters with stride 1 and the same padding, and 3 fully connected layers. Before the max pooling layer at 2x2 with stride 2, two convolutional layers are passed through. Thus, valuable information is preserved despite the reduction in spatial dimensionality. The activation function used after the convolutional and fully connected layers is ReLU [8]. Softmax is used in the final layer and outputs probabilities for each category in the bin [7]. In other words, multi-class classification.

To leverage the knowledge gained from training on large-scale datasets and accelerate the training process, a VGG16 model pre-trained on the ImageNet dataset was used as a baseline (pre-trained model) [11], [12], [14]. In this case, efficiency was also a goal. VGG16 has powerful generic feature extraction, so it is preferable to freeze it first. The model only needs to be adjusted in the final fully connected layer to function on bin classification. This way, the model can learn relevant patterns in the dataset. This approach is desirable because it can reduce training time and required computational resources [6], [16].
The model training process was carried out with the following configuration:
1. Optimizer: The Adam optimizer was used to optimize the model [7], with an initial learning rate set at 0.001.
2. Loss Function: Categorical cross-entropy is used as the loss function [12], which is suitable for multi-class classification tasks.
3. Callbacks: To monitor and control the training process, callbacks such as ModelCheckpoint and EarlyStopping are implemented. ModelCheckpoint stores the best model weights based on performance on the validation set (e.g., val_loss), while EarlyStopping stops training if validation performance does not improve after a certain number of epochs, preventing overfitting.
4. Epochs and Batch Size: The number of epochs and batch size were determined experimentally to achieve optimal convergence and efficient resource utilization.

2.3. Model Conversion to TensorFlow.js

The VGG16 model, which had been trained and saved in HDF5 Keras (.h5) format, was converted to a format compatible with TensorFlow.js [17]. This was done using the tensorflowjs_converter command-line tool. The conversion resulted in a model.json file and several binary shard files containing the model weights. The VGG16 model is very large, containing approximately 138 million parameters[3], and its weight files are approximately 100 MB. Even after conversion to TensorFlow.js format, the separate binary files are expected to remain large. This large file size significantly impedes the initial download time of the PWA and memory consumption in the user's browser. This significantly impacts the user experience with the PWA, which is expected to be fast and highly responsive. Therefore, simply converting the model is not sufficient; substantial model optimization strategies[18] need to be implemented before or after conversion to make the VGG16 model practically suitable for PWA deployment.

2.4. Progressive Web App (PWA) Development

Figure 1 illustrates the overall system flowchart, demonstrating the sequential process from user image capture through preprocessing, model inference, and result display. Figure 2 provides a detailed view of the client-side processing logic, emphasizing the conditional flow for cache utilization and the fallback mechanism for network retrieval when cached models are unavailable. These diagrams highlight

the system's ability to function independently of server connectivity once the initial model is cached.

To provide a complete overview of the system flow, the application's overall architecture and logical flow diagram is presented below:

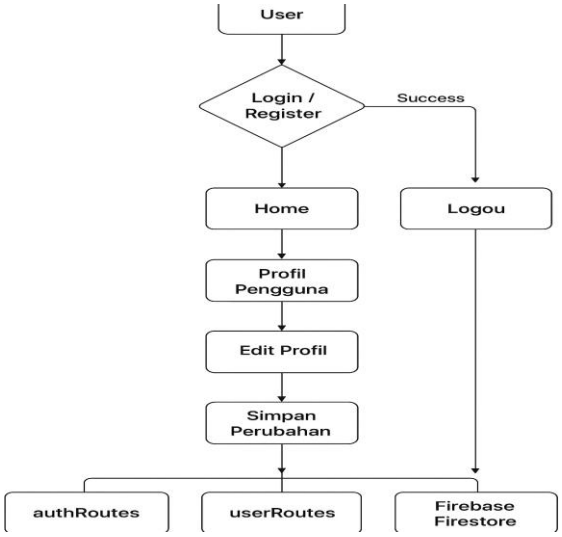

Figure 1. Flowchart of a PWA-Based Waste Classification System

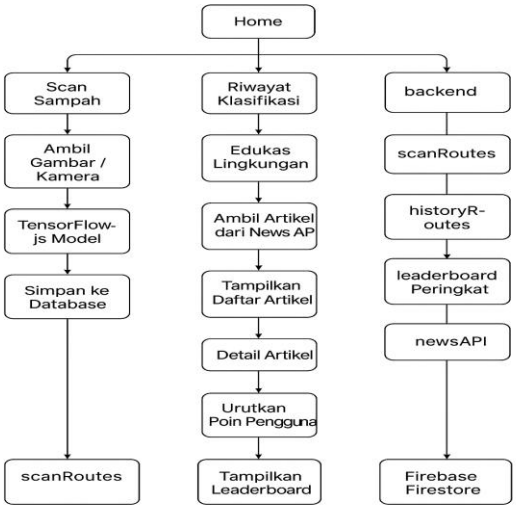

Figure 2. Flowchart of a PWA-Based Waste Classification System

To ensure the application functions offline and loads the large model efficiently, a custom Service Worker strategy is implemented. Listing 2 shows how the Service Worker intercepts network requests and serves cached assets, which is critical for the PWA performance

Listing 2. Service Worker Caching Strategy

```
// Service Worker (sw.js)
self.addEventListener('fetch', (event) => {
  // Intercept request
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached file if available
        if (cachedResponse) {
          return cachedResponse;
        }
        // Otherwise, fetch from network
        return
fetch(event.request).then((networkResponse)  =>
{
          // Cache the new file for future use
          return              caches.open('vgg16-cache-
v1').then((cache) => {
            cache.put(event.request,
networkResponse.clone());
            return networkResponse;
          });
        });
      })
  );
});
```

To ensure the system runs efficiently on mobile devices with limited resources, the application architecture is designed to prioritize client-side processing. Figure 3 illustrates the three-layer system architecture: the client-side processing unit (React UI and TensorFlow.js), the service layer (Service Worker and IndexedDB for offline capability), and the cloud backend (Firebase for optional data synchronization). As depicted, the VGG16 model executes entirely on the user's device using the WebGL backend for GPU acceleration, eliminating server dependency during inference.

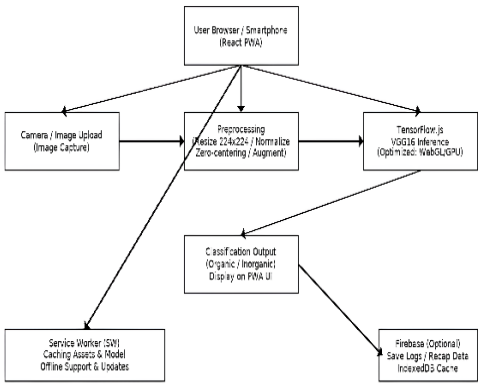

Figure 3. System Block Diagram of the Proposed PWA Waste Classification.

As shown in Figure 3, the system consists of three main layers: the client-side processing unit (React UI and TensorFlow.js), the service layer (Service Worker and IndexedDB for offline capability), and the cloud backend (Firebase). The VGG16 model is executed directly on the user's device using the WebGL

backend for GPU acceleration. To ensure optimal performance on mobile devices, the system explicitly configures TensorFlow.js to use the WebGL backend with specific memory optimizations, as shown in Listing 3.

Listing 3. WebGL Backend Configuration for GPU Acceleration

```
// Initialize backend for better performance
        async function initBackend() {
          try {
            if (tf.getBackend() !== 'webgl') {
              await tf.setBackend('webgl');
              await tf.ready();

              // Enable float16 textures for memory efficiency
              if (tf.ENV.getBool('WEBGL_VERSION') === 2) {

tf.env().set('WEBGL_FORCE_F16_TEXTURES', true);

tf.env().set('WEBGL_DELETE_TEXTURE_THRESHOLD', 0);
              }
            }
          } catch (err) {
            console.warn('Fallback to CPU:', err);
            await tf.setBackend('cpu');
          }
        }
```

The application was developed as a Progressive Web App (PWA) using React for the frontend, Hapi as the backend API, and Firebase for data storage. The manifest.json file is used to define the application name, icon, start URL, standalone view mode, and theme and background colors. A service worker is registered to support asset caching, enable offline access, and speed up loading. The VGG16 waste classification model in TensorFlow.js format is loaded directly on the frontend asynchronously, with priority fetching from the IndexedDB cache for optimal performance. Images captured by the camera or uploaded by the user are processed directly on the client side (resize to 224x224, normalize, and adjust colors) before inference is performed by the model. This process runs on the main thread or a web worker to maintain a responsive UI, and the classification results are displayed in real time.

III.RESULT AND DISCUSSION

3.1. Presentation of Results

To measure the performance of the classification model, standard evaluation metrics were calculated based on the confusion matrix values. The formulas used are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$Precision = \frac{TP}{TP + FP} \times 100\%$$

$$Recall = \frac{TP}{TP + FN} \times 100\%$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where TP represent True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives.

The retrained VGG16 model achieved an overall accuracy of 0.94 on the test set. Evaluation of each category using precision, recall, and F1-score metrics yielded the following results:

Table 1: Comparison of VGG16 Model Waste Classification Performance

Waste Category	Precision (%)	Recall (%)	F1-Score (%)
Inorganic	95	96	95
Organic	94	93	94
Total Accuracy			94

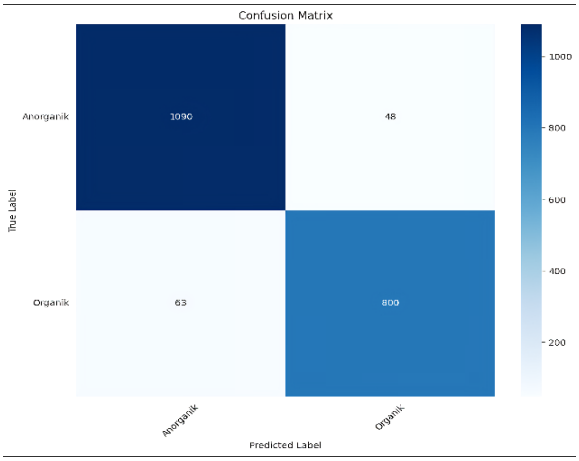

Figure 4: Confusion Matrix

Figure 4 presents a confusion matrix detailing the performance of the VGG16 model on 2,001 test datasets. This matrix shows that the model successfully identified 1,090 images as 'Inorganic' (True Positives) correctly and 800

images as 'Organic' (True Positives). There were two main types of errors: 63 'Organic' images were misclassified as 'Inorganic' (False Positives), and 48 'Inorganic' images were misclassified as 'Organic' (False Negatives). This relatively small number of prediction errors (a total of 111 out of 2001) confirms the model's high accuracy of 94%. This data also aligns with Table 1, which shows a slightly higher recall for the Inorganic class (96%) than for the Organic class (93%), indicating the model is slightly more reliable in recognizing inorganic waste.

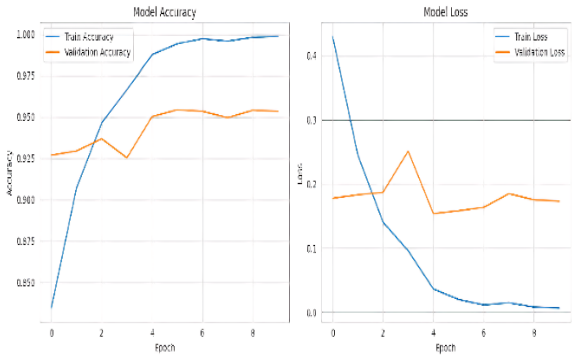

Figure 5: Accuracy Graph

Figure 5 displays a graph of the model's training history, plotting the accuracy (left) and loss (right) metrics for the training data (blue line) and validation data (orange line) over 15 epochs. The accuracy graph shows that both training accuracy (accuracy) and validation accuracy (val_accuracy) have steadily increased. Concurrently, the loss graph shows a consistent decline for both data sets (loss and val_loss). Importantly, there is no significant gap or divergence between the training and validation curves. This demonstrates that the model does not suffer from severe overfitting and has good generalization ability when applied to new data.

In PWA Inference Performance, the 100 MB VGG16 model in TensorFlow.js format affects the initial download and loading time of the model. For a resolution of 224x224 pixels, the inference time is approximately 109.79 ms, but for a resolution of 512x512, it jumps to 18 seconds.

Example Application Interface

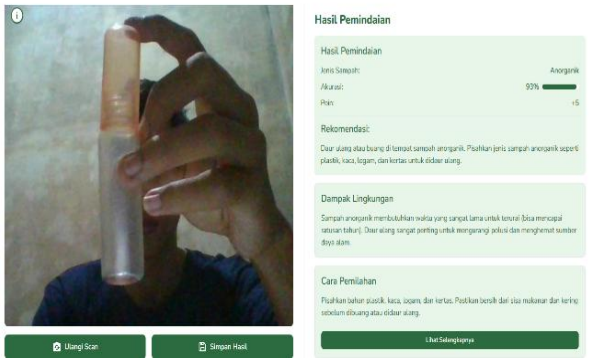

Figure 6: PWA Interface for Automatic Waste Classification

Figure 6 showcases the responsive PWA interface during real-time waste classification. The clean, intuitive design enables users to capture or upload waste images and receive instant classification results with confidence scores. The interface's simplicity ensures accessibility for diverse user demographics, from students to waste management practitioners, supporting the system's educational and practical objectives.

3.2. Discussion

VGG16 Model Performance in Waste Classification: VGG16 demonstrates adequate classification performance with a total accuracy of 94%. This model performs quite well in classifying organic and inorganic waste categories. This indicates that the VGG16 convolutional architecture has captured distinctive visual features such as texture and dominant color. The performance degradation in garbage classification is likely due to the uneven number of samples in the training dataset and the similarity between categories. Deep learning models are powerful, but they still require the quality and diversity of the dataset [4].

Table 2. Comparison with Previous Waste Classification Systems

Study	Model	Acc. (%)	Deployment	Internet	Inst.	Device
This Work	VGG16	94.0	PWA	No	No	Any
[3]	Ens.	97.2	Server	Yes	Yes	Mob.
[6]	WNet	91.8	IoT/Edge	No	HW	Ded.
[8]	MNetV2	89.3	Mob. App	No	Yes	Mob.
[7]	VGG16	95.1	Server	Yes	Yes	Mob.
[16]	Custom	88.5	Edge	No	HW	Ded.

Notes: Acc.=Accuracy; Inst.=Installation; Ens.=Ensemble CNN; WNet=WasteNet;

MNetV2=MobileNetV2; HW=Hardware Setup; Mob.=Mobile; Ded.=Dedicated Device.

As shown in Table 2, the proposed system achieves competitive accuracy (94%) while uniquely combining client-side processing with zero installation requirements. While Nahiduzzaman et al. [3] achieved higher accuracy (97.2%), their server-dependent architecture introduces latency and privacy concerns from continuous data transmission. In contrast, our client-side approach processes images locally, eliminating these issues at the cost of a modest 3.2% accuracy reduction. Compared to MobileNetV2-based solutions [8], which prioritize model lightness over accuracy, our VGG16 implementation demonstrates that larger, more accurate models can be practically deployed in browsers through strategic optimizations (GPU acceleration, caching). The 4.7% accuracy improvement over MobileNetV2 justifies the additional computational requirements, particularly for educational and high-stakes sorting applications where classification reliability is paramount. Unlike IoT-based edge systems [6], [19] that require specialized hardware setup, the PWA approach leverages ubiquitous smartphone browsers, dramatically lowering the barrier to entry for waste management education and community engagement. This approach also offers a more accessible alternative to dedicated IoT hardware solutions, such as the LoRa-GPS smart bins [18]. This accessibility advantage outweighs the 2.2% accuracy difference with hardware-optimized WasteNet [6], especially in resource-constrained settings where purchasing dedicated devices is impractical. The key contribution of this work lies in demonstrating that high-accuracy CNN models (138M parameters) can be effectively deployed at the edge through web technologies, achieving 94% accuracy with full offline capability---a combination not demonstrated in prior browser-based solutions [16]. This validates PWA + TensorFlow.js as a viable platform for deploying computationally intensive AI models in socially impactful applications.

The implementation of a waste classification system has the potential to automate the manual process of waste sorting [3], [7]. This may raise concerns about job losses. However, it creates new opportunities in the form of digitalization in the waste management sector, such as retraining workers to become educators or operators in the waste sorting process. The integration of AI into public applications must address fairness and transparency. For example, if a model is more accurate with certain types of waste, this could lead to classification bias [3]. Furthermore, classification results, along with user data, must be handled within strict privacy policy constraints. While the goal of this application is to manage waste more effectively, there are still drawbacks in terms of energy impact due to indirect energy use from training and inference. Implementing a lightweight and efficient model is a crucial first step towards environmental sustainability in the digital realm [8], [15].

This system is accessible from various devices and network conditions because it adopts the PWA (Progressive Web App) principle. Even without an internet connection, users can still utilize the classification feature. Push notifications and homescreen placement increase long-term user engagement. The app's widespread use of low- to mid-range devices and responsiveness contribute to the reduction of the digital divide. These features align with the mission of educating and engaging the public about responsible waste management, while simultaneously making this technology an inclusive and sustainable solution.

IV.CONCLUSION

This research successfully developed an accessible automatic waste classification system by integrating the VGG16 CNN model into a Progressive Web Application, demonstrating that high-accuracy deep learning models can be effectively deployed client-side for offline waste classification. The system achieved 94% accuracy on a validated dataset of 10,365 images through strategic implementation of TensorFlow.js with GPU acceleration (WebGL backend) and robust caching mechanisms (Service Worker and IndexedDB), confirming that PWA technology can support computationally intensive AI applications without sacrificing accessibility or offline functionality. The main contribution of this work is providing a replicable framework for deploying large-scale CNN models (138M parameters) at the edge through web browsers, eliminating installation barriers and server dependencies that limit existing solutions. This approach directly addresses the research gap in client-side waste classification by demonstrating that accuracy need not be sacrificed for accessibility—our VGG16 implementation outperformed lightweight alternatives while maintaining universal browser compatibility. The primary limitation remains the 100 MB model size causing initial load latency on slow networks. Future work should explore: (1) advanced compression techniques (quantization, pruning, knowledge distillation) to reduce model size without significant accuracy loss, (2) lighter architectures like EfficientNet for comparative evaluation, and (3) dataset expansion to include

hazardous waste and underrepresented recyclable categories, enhancing system robustness and reducing classification bias. These improvements will further strengthen PWA-based AI deployment as a sustainable, inclusive solution for environmental education and waste management practice.

REFERENCE

[1] B. Fang *et al.*, "Artificial intelligence for waste management in smart cities: a review," Aug. 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s10311-023-01604-3.

[2] D. Ziouzios, N. Baras, V. Balafas, M. Dasygenis, and A. Stimoniaris, "Intelligent and Real-Time Detection and Classification Algorithm for Recycled Materials Using Convolutional Neural Networks," *Recycling*, vol. 7, no. 1, Feb. 2022, doi: 10.3390/recycling7010009.

[3] M. Nahiduzzaman *et al.*, "An automated waste classification system using deep learning techniques: Toward efficient waste recycling and environmental sustainability," *Knowl Based Syst*, vol. 310, Feb. 2025, doi: 10.1016/j.knosys.2025.113028.

[4] F. R. Sayem *et al.*, "Enhancing waste sorting and recycling efficiency: robust deep learning-based approach for classification and detection," *Neural Comput Appl*, Feb. 2024, doi: 10.1007/s00521-024-10855-2.

[5] A. Arishi, "Real-Time Household Waste Detection and Classification for Sustainable Recycling: A Deep Learning Approach," *Sustainability (Switzerland)*, vol. 17, no. 5, Mar. 2025, doi: 10.3390/su17051902.

[6] G. White, C. Cabrera, A. Palade, F. Li, and S. Clarke, "WasteNet: Waste Classification at the Edge for Smart Bins," Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.05873

[7] M. I. B. Ahmed *et al.*, "Deep Learning Approach to Recyclable Products Classification: Towards Sustainable Waste Management," *Sustainability (Switzerland)*, vol. 15, no. 14, Jul. 2023, doi: 10.3390/su151411138.

[8] L. Yong, L. Ma, D. Sun, and L. Du, "Application of MobileNetV2 to waste classification," *PLoS One*, vol. 18, no. 3 March, Mar. 2023, doi: 10.1371/journal.pone.0282336.

[9] H. Zheng and Y. Gu, "Encnn-upmws: Waste classification by a CNN ensemble using the UPM weighting strategy," *Electronics (Switzerland)*, vol. 10, no. 4, pp. 1–21, Feb. 2021, doi: 10.3390/electronics10040427.

[10] L. Stephen Pieters, "DEVELOPMENT OF AUTOMATIC WASTE CLASSIFICATION SYSTEM USING CNN BASED DEEP LEARNING TO SUPPORT SMART WASTE MANAGEMENT PENGEMBANGAN SISTEM KLASIFIKASI SAMPAH OTOMATIS MENGGUNAKAN DEEP LEARNING BERBASIS CNN UNTUK MENDUKUNG SMART WASTE MANAGEMENT," vol. 10, no. 1, p. 2025.

[11] A. Gaurav *et al.*, "Smart waste classification in IoT-enabled smart cities using VGG16 and Cat Swarm Optimized random forest," *PLoS One*, vol. 20, no. 2 February, Feb. 2025, doi: 10.1371/journal.pone.0316930.

[12] J. D. Ortiz-Mata, X. J. Oleas-Vélez, N. A. Valencia-Castillo, M. del R. Villamar-Aveiga, and D. E. Dáger-López, "Comparison of Vertex AI and Convolutional Neural Networks for Automatic Waste Sorting," *Sustainability (Switzerland)*, vol. 17, no. 4, Feb. 2025, doi: 10.3390/su17041481.

[13] C. Shi, C. Tan, T. Wang, and L. Wang, "A waste classification method based on a multilayer hybrid convolution neural network," *Applied Sciences (Switzerland)*, vol. 11, no. 18, Sep. 2021, doi: 10.3390/app11188572.

[14] Z. Md, A. Amin, N. Sami, and R. Hassan, "An Approach of Classifying Waste Using Transfer Learning Method," 2021.

[15] W. Qiu, C. Xie, and J. Huang, "An improved EfficientNetV2 for garbage classification," Mar. 2025, [Online]. Available: http://arxiv.org/abs/2503.21208

[16] X. Li and R. Grammenos, "A Smart Recycling Bin Using Waste Image Classification At The Edge," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.00448

[17] D. Smilkov *et al.*, "TENSORFLOW.JS: MACHINE LEARNING FOR THE WEB AND BEYOND," 2019.

[18] N. C. A. Sallang, M. T. Islam, M. S. Islam, and H. Arshad, "A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment," *IEEE Access*, vol. 9, pp. 153560–153574, 2021, doi: 10.1109/ACCESS.2021.3128314.

[19] I. Fanani and R. Rianto, "Improving Online Exam Verification with Class-Weighted and Augmented CNN Models," *Jurnal Teknologi Informasi dan Terapan (J-TIT)*, vol. 11, no. 2, pp. 91-98, 2024.

[20] F. Ramadhani *et al.*, "Klasifikasi Suara Paru Normal Dan Abnormal Berbasis Algoritma CNN (Convolutional Neural Network)," *Jurnal Teknologi Informasi dan Terapan (J-TIT)*, vol. 11, no. 1, pp. 15-20, 2024.

EKA SETYABUDI is currently an undergraduate student in the Informatics Department at the Universitas PGRI Semarang, Semarang, Indonesia. He is in the seventh semester of the bachelor program. His research interests primarily include machine learning, data mining, and the application of artificial intelligence in various fields. He can be contacted at email: 22670144@upgris.ac.id

NOORA QOTRUN NADA is a Lecturer in the Informatics Study Program and currently serves as the Secretary of the Program Study at the Universitas PGRI Semarang, Semarang, Indonesia. She obtained her Bachelor of Engineering (S.T.) and Master of Engineering (M.Eng.) degrees. Her primary research interests include software engineering, information systems, and the application of technology in education. She can be contacted at email: noora@upgris.ac.id

MEGA NOVITA is an Associate Professor in the Informatics Study Program at the Universitas PGRI Semarang, Semarang, Indonesia. She obtained the S.Si. degree in mathematics from Satya Wacana Christian University (UKSW) in 2009, and the M.Si. degree in biology from UKSW in 2011. She later received the M.Nat.Sc. degree in chemistry from Kwansei Gakuin University (KGU), Japan, in 2012, and earned the Doctor of Science (Dr.Sc.) degree from the same university in 2015. Her interdisciplinary research background focuses on computational science, theoretical investigation of electronic structures, and data analysis, particularly their application in materials science and computational modeling. She also conducted postdoctoral research at Chonbuk National University (CBNU), South Korea. She can be contacted at email: mega@upgris.ac.id