



Research on Lightweight Garbage Classification Algorithm Based on Logistic Regression and Its Real-Time System Implementation

Changhang Ren
Dongguan University of Technology
Dongguan, China
173454722@qq.com

Peidong Lai*
School of Business
Dongguan City University
Dongguan, China
laipeidong@dgcu.edu.cn

Zheyu Xu
Dongguan University of Technology
Dongguan, China
1580766946@qq.com

Abstract

In recent years, waste sorting has become a key focus in urban environmental management worldwide, and its effectiveness depends on advanced classification algorithms and practical deployment technologies. However, due to the limited computational power and storage resources of embedded devices, applying efficient classification algorithms in these resource-constrained scenarios remains challenging. This study aims to address this issue by optimizing the Logistic Regression model for lightweight application to meet the requirements of embedded waste sorting systems. Through parameter pruning and model quantization methods, the storage and computational costs of the model were significantly reduced. The models, before and after the lightweight optimization, were deployed on both PCs and embedded devices in the experiments, and their performance in terms of classification accuracy and inference time was compared. The results showed that after optimization, the inference time on embedded devices was significantly reduced to 19.86 milliseconds, with a slight decrease in classification accuracy, but still within a reasonable range. Moreover, the improvement in inference time was statistically significant ($p < 0.05$). These findings indicate that lightweight optimization can enhance the real-time performance of embedded systems while maintaining classification performance, showing broad application potential, especially in waste sorting scenarios within Internet of Things (IoT) environments.

CCS Concepts

• Computer systems organization; • Embedded and cyber-physical systems; • Embedded systems;

Keywords

Lightweight optimization, Logistic regression, Garbage classification, Embedded system

*corresponding author

ACM Reference Format:

Changhang Ren, Peidong Lai, and Zheyu Xu. 2024. Research on Lightweight Garbage Classification Algorithm Based on Logistic Regression and Its Real-Time System Implementation. In *2024 International Symposium on Integrated Circuit Design and Integrated Systems (ICDIS 2024)*, November 22–24, 2024, Xiamen, China. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3702191.3703360>

1 Introduction

With the increase in global waste production, the importance of waste management and classification in modern urban governance has become increasingly prominent. Especially in the context of accelerated urbanization and changing lifestyles, the significant growth in the variety and volume of waste has made resource wastage and environmental pollution critical issues [1]. Intelligent waste sorting systems are considered one of the effective means to solve problems related to waste recycling and environmental pollution. However, most existing machine learning models rely on high-performance devices, limiting their application in resource-constrained embedded environments [2]. Embedded devices, with their advantages of low power consumption and low cost, are suitable for intelligent waste classification in the context of the Internet of Things (IoT). However, their limited resources make it difficult for traditional models to meet the demands of real-time and efficient processing. Therefore, optimizing lightweight models to enable the Logistic Regression algorithm to function within embedded systems while maintaining reasonable classification performance has become an important research direction. This study optimizes the Logistic Regression model through parameter pruning and model quantization, enabling its operation on embedded devices with high real-time performance and reasonable accuracy. The results demonstrate that the optimized model significantly reduces storage requirements and computational complexity while maintaining effective classification performance in embedded environments, providing strong support for the practical application of intelligent waste classification systems in IoT.

2 Research Foundation

The theoretical foundation of this study primarily stems from the Logistic Regression model in machine learning and lightweight optimization theory. Logistic Regression is one of the fundamental algorithms widely applied in classification tasks, possessing good generalization capability and relatively low computational complexity, making it suitable for waste classification tasks in embedded device environments [3]. Data augmentation techniques also play

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDIS 2024, November 22–24, 2024, Xiamen, China

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1822-9/24/11

<https://doi.org/10.1145/3702191.3703360>

a key role in this study. By increasing data diversity, these techniques can effectively improve the model's generalization ability and reduce overfitting, which is particularly important for small-scale, self-built waste datasets. In terms of model optimization, this study applies parameter pruning and quantization techniques, which have been proven in recent years to significantly reduce computational resource consumption and improve the real-time processing capabilities of embedded devices in embedded artificial intelligence research [4]. Additionally, the study adopts a Real-Time Operating System (RTOS) and task scheduling framework, which provide reliable support for the stable operation of the model on resource-constrained embedded platforms through real-time optimization methods. By integrating these foundational theories, this study provides a systematic solution for achieving real-time waste classification in embedded environments.

3 Materials and Methods

3.1 Experimental environment and tools

3.1.1 Hardware Equipment. In the experiment, a desktop computer equipped with an Intel Core i7-10700K processor (8 cores, 16 threads, 3.8 GHz), 32GB of RAM, and a 1TB NVMe SSD was used for algorithm development and testing. A Raspberry Pi 4 Model B (with 4GB RAM, 1.5 GHz ARM Cortex-A72 quad-core processor) was used for embedded experiments to verify the performance of the lightweight waste classification algorithm. An NVIDIA RTX 3070 graphics card (8GB VRAM) accelerated large-scale matrix computations during the model training process. An RGB camera (1080P) and a weight sensor were utilized for real-time acquisition of waste images and weight data to assist in classification.

3.1.2 Software Tools. The experiment was conducted using a desktop computer running the Ubuntu 20.04 LTS operating system, and a Raspberry Pi device running Raspberry Pi OS (based on Debian Buster). The development languages included Python 3.8 for implementing the Logistic Regression algorithm, C++ for enhancing the real-time performance of the embedded system, and Shell scripts for automating tasks. Machine learning and image processing libraries used included Scikit-learn 0.24.2, OpenCV 4.5.1, Pandas, and Numpy. FreeRTOS 10.4.3 was employed on the Raspberry Pi to manage real-time tasks. SQLite 3.34 was used for data storage, and the classification results were transmitted in real-time using the MQTT protocol.

3.2 Dataset construction and preprocessing

3.2.1 Data Collection. By setting up a smart trash bin equipped with an RGB camera in the experimental environment, waste images were collected, covering six categories of waste. The camera was positioned approximately 30 cm above the upper edge of the trash to ensure the images captured were clear and unobstructed. A total of 6,000 images were collected, with approximately 1,000 images for each category.

3.2.2 Data Augmentation. To enhance the generalization ability of the classifier, data augmentation methods were applied to expand the self-built dataset. The following data augmentation techniques were used for data expansion [5]: (1) Rotation: Images were rotated by 10°, 20°, and 30°, respectively, to ensure sufficient angular

variation. New image samples were generated for each rotation method. (2) Scaling and Translation: Each image was scaled by 5% to 20% and shifted horizontally or vertically (specifically by 10 to 30 pixels) to enhance spatial invariance. (3) Brightness Adjustment: The brightness of the images was modified within a 20% range, both increasing and decreasing the brightness, to ensure the model could adapt to environments with varying brightness levels. (4) Flipping: Horizontal and vertical flips were performed on the images, creating reversed image samples to increase dataset diversity.

3.2.3 Data Preprocessing. (1) Grayscale Conversion: Using the OpenCV library, RGB three-channel images were converted to grayscale to reduce computational resource requirements while retaining the main structural features. (2) Normalization: All image pixel values were normalized to the range of [0, 1] to reduce the impact of differences in data magnitude on training. (3) Feature Extraction and Dimensionality Reduction: The Sobel operator was used to extract edge features from the images, and Principal Component Analysis (PCA) was applied to reduce the features to 64 dimensions, reducing redundant features and improving computational efficiency.

3.3 Logistic regression model design

3.3.1 Model Structure and Optimization. (1) Logistic Regression Model: Based on the multi-class classification problem, the "One-vs-Rest" (OVR) strategy is adopted, converting the six-class garbage classification problem into multiple binary classification problems for processing [6]. The Logistic Regression model is used to calculate the probability of each class and applies the Sigmoid function to map the input features. The mathematical formula is as follows:

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_0 X_0 + \beta_1 X_1 + \dots + \beta_n X_n)}} \quad (1)$$

Where "y" represents the probability that the model outputs a certain class, and $\beta_0, \beta_1, \dots, \beta_n$ are the model's bias and weight parameters.

(2) Loss Function and Regularization: The logarithmic loss function (Log Loss) is used as the objective function, with the addition of an L2 regularization term and a regularization coefficient of 0.01 to prevent overfitting. The Log Loss function measures model performance by calculating the difference between predicted and actual values. Its mathematical formula is as follows:

$$L = \frac{1}{m} \sum_{i=1}^m [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \sum_{j=1}^n \beta_j^2 \quad (2)$$

Where "L" is the loss function, " y_i " is the actual label, " \hat{y}_i " is the predicted probability, "m" is the number of samples, and " λ " is the regularization coefficient used to suppress overfitting.

(3) Model Training: The training is performed using the Batch Gradient Descent method with a batch size set to 32, a maximum of 100 iterations, and a learning rate of 0.001. Through iterative optimization, the model parameters are adjusted to minimize the loss function until convergence is achieved.

3.3.2 Model Lightweight Optimization. (1) Parameter Pruning: Parameter pruning is applied to the trained Logistic Regression model by removing parameters with absolute values below a set threshold

(0.005), reducing the model size to 70% of the original. Parameter pruning is a method used to reduce model complexity by removing weights with lower importance. Specifically, the pruning rule is as follows:

$$\beta_j = 0, \text{ if } |\beta_j| < \epsilon \quad (3)$$

Where “ ϵ ” is the set threshold. Pruning can significantly reduce the number of model parameters, thereby lowering storage and computation costs while retaining the key parameters to maintain model performance as much as possible.

(2) Model Quantization: The model uses 8-bit integer quantization (INT8) to convert floating-point parameters into low-precision integers, reducing memory usage. The symmetric quantization method is used during the quantization process to maintain simplicity and precision in computation [7]. Model quantization reduces storage and computational complexity by mapping floating-point parameters to low-precision integers. In this study, 8-bit integer quantization (INT8) is used, which linearly scales the weights into integers:

$$q = \text{round} \left(\frac{w - w_{\min}}{\Delta} \right) \quad (4)$$

Where “ q ” is the quantized integer, “ w ” is the original floating-point parameter, “ w_{\min} ” is the minimum value of the parameter, and Δ is the quantization interval (i.e., the scaling factor). Symmetric quantization maintains model accuracy while reducing computational complexity.

3.4 Real-Time system integration

(1) Task Division and Scheduling: Using the FreeRTOS real-time operating system, the waste classification process was divided into several independent tasks: image acquisition, preprocessing, inference, and result transmission. Task priorities were set based on CPU and memory usage, as recorded by the Linux performance monitoring tool htop, to assess resource consumption [8]. (2) System Latency: The overall system latency, measured from image acquisition to the transmission of classification results, was required to be within 100 milliseconds.

3.5 Statistical analysis methods

A one-way analysis of variance (ANOVA) was conducted to compare the Logistic Regression model’s performance before and after lightweight optimization under different experimental conditions, aiming to determine whether the lightweight optimization had a significant impact on the model’s performance. The F-value and p-value were calculated, and if the p-value was less than 0.05, it was concluded that the optimization had a statistically significant effect on the model’s performance. To ensure result stability, a 95% confidence interval was calculated for each experimental metric based on the results of cross-validation, thereby enhancing the reliability of the evaluation. To further assess the model’s classification performance, ROC curves were plotted for both the pre- and post-optimization models, and their AUC values were calculated. The comparison of AUC values was used to evaluate the impact of lightweight optimization on classification ability. In the resource utilization analysis, descriptive statistics were used to calculate the mean and variance of CPU and memory usage, in order to assess



Figure 1: Example of Waste Image Samples Collected by the Camera: A: Paper, B: Plastic, Glass, C: Metal, D: Kitchen Waste, E: Hazardous Waste.

the differences in resource consumption of the model before and after lightweight optimization within the embedded system [9].

4 Application Validation and Simulation Analysis

4.1 Dataset presentation and effects of data augmentation

In the experiment, images of waste samples were collected using an RGB camera to display the appearance characteristics of different types of waste. The image samples covered six categories of waste (A: paper, B: plastic, glass, C: metal, D: kitchen waste, E: hazardous waste), with representative images shown in Figure 1. The images for each waste category were collected in a standardized manner, ensuring consistency in the capture angle, resolution (1080P), and lighting conditions. Figure 1 clearly illustrates the visual differences between various types of waste under standardized collection conditions, which aids the subsequent classification algorithm in effectively identifying waste categories and extracting relevant features.

After data augmentation, different augmentation methods had varying impacts on the model’s classification accuracy (Figure 2). A t-test revealed that, compared to the original dataset, classification accuracy significantly improved with rotation, scaling, brightness adjustment, and flipping augmentations ($p < 0.05$), while the difference with translation augmentation was not significant ($p = 0.071$). These results indicate that certain data augmentation techniques effectively enhanced the model’s generalization capability. The

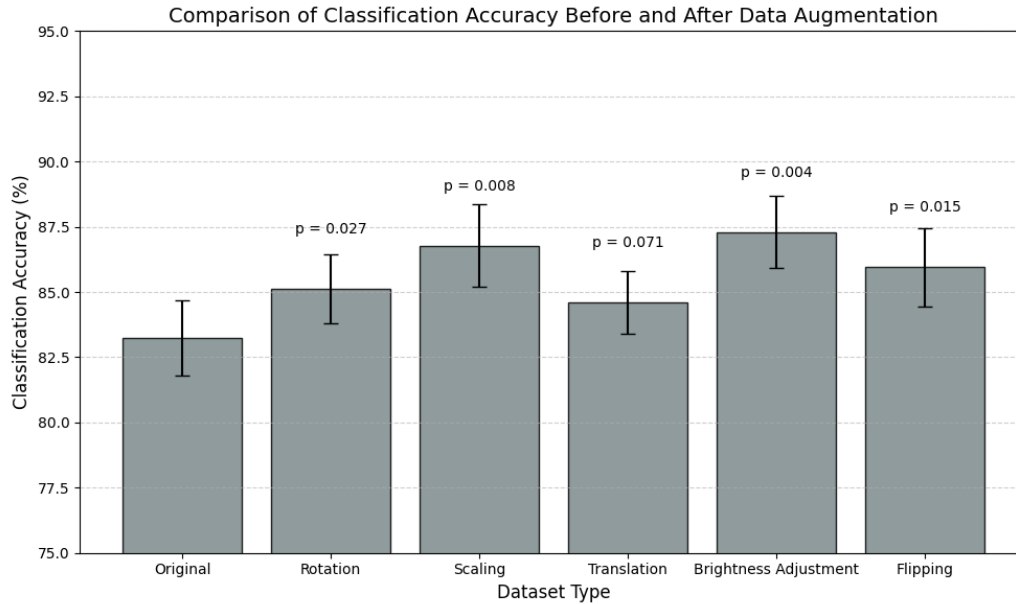


Figure 2: Comparison of Model Classification Accuracy Before and After Data Augmentation.

Table 1: Parameter count, storage size, and classification accuracy of the Logistic Regression model before and after lightweight optimization

Metric	Original Model	Lightweight Optimized Model
Number of parameters (million)	1.5	1.05
Storage size (MB)	12.8	8.3
Classification accuracy (PC, %)	86.45 ± 1.32	85.23 ± 1.45
Classification accuracy (Embedded, %)	82.38 ± 1.45	80.95 ± 1.58
Paired t-value (PC)	-	3.162
p-value (PC)	-	0.006
Paired t-value (Embedded)	-	2.768
p-value (Embedded)	-	0.015

error bars represent standard error (SE), further illustrating the variability in classification accuracy across groups. The fluctuation in accuracy after augmentation was relatively smaller, reflecting an improvement in the model's stability due to the augmentation methods.

4.2 Model training and lightweight optimization effects

During the model training process, the loss value gradually decreased and stabilized, confirming the model's convergence and generalization capability (Figure 3A). The lightweight optimization significantly reduced the number of model parameters and storage size, with only a slight decrease in classification accuracy (Table 1). Paired t-tests on both PC and embedded devices showed significant results ($p = 0.006$, $p = 0.015$, respectively). The inference time decreased from 12.58 ms to 9.45 ms on the PC and from 27.32 ms to 19.86 ms on the embedded device, both showing statistically

significant improvements (PC: $p = 0.002$, embedded: $p = 0.001$), indicating that the lightweight optimization effectively improved the model's inference efficiency (Figure 3B).

4.3 Real-Time performance evaluation of embedded system

In the real-time performance evaluation of the embedded system, the latency of each step in the waste classification task was measured. The results showed that the average time consumption for image acquisition, image preprocessing, inference, and result transmission were 32.47 milliseconds, 21.38 milliseconds, 29.64 milliseconds, and 14.23 milliseconds, respectively. The overall average system latency was 97.72 milliseconds (Figure 4). These results indicate that the time consumption for each task step was within a reasonable range, and the overall system latency remained below 100 milliseconds, meeting the requirements for real-time applications in embedded systems. This time distribution ensures that

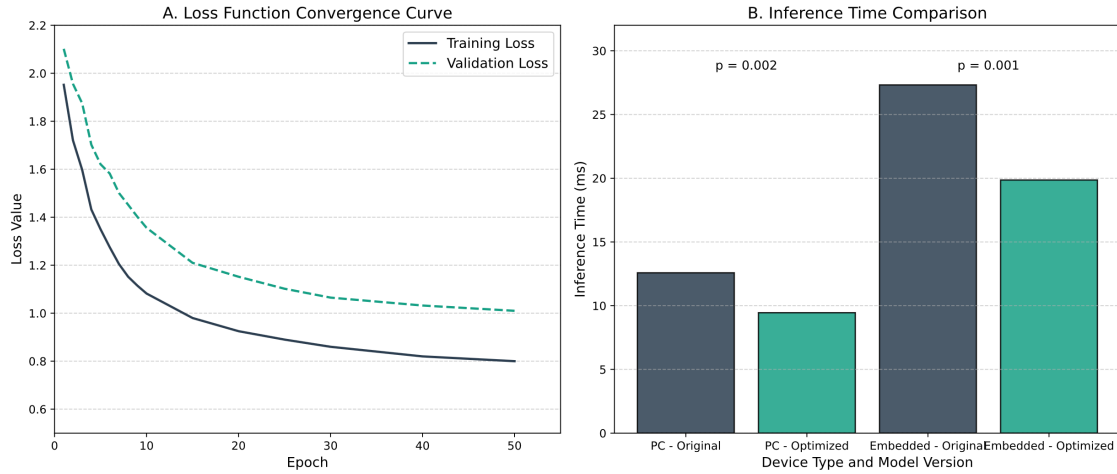


Figure 3: Model Training Process and Lightweight Optimization. A: Loss Function Convergence Curve during the Training Process (Logistic Regression); B: Comparison of Inference Time Before and After Lightweight Optimization of the Logistic Regression Model.

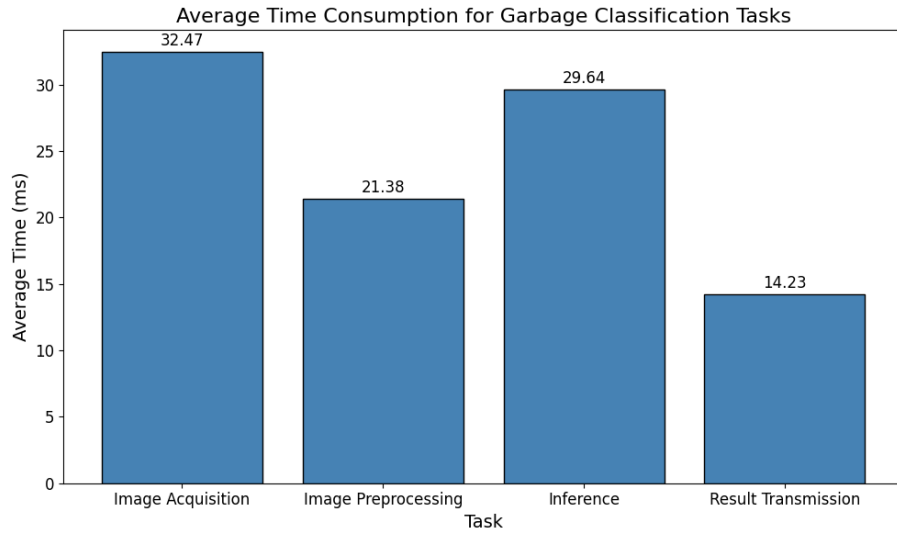


Figure 4: Task Time Consumption for Each Step in the Waste Classification Task.

the system's response speed can support the real-time demands of waste classification.

4.4 Significance analysis of model performance

In the significance analysis of the performance of the Logistic Regression model before and after lightweight optimization, one-way ANOVA showed significant differences in the performance of the lightweight optimization under different experimental conditions. Specifically, classification accuracy and inference time exhibited significant changes in both PC and embedded device environments (Table 2). Notably, the inference time on embedded devices showed particularly prominent significance, with an F-value of 10.331 and a

p-value of 0.002, indicating that lightweight optimization had a substantial impact in resource-constrained environments. These results confirm that the lightweight optimization significantly improved model performance, especially in real-time and resource-limited conditions.

To assess the impact of lightweight optimization on model performance, the 95% confidence intervals for classification accuracy and inference time were calculated. The results showed that, in both PC and embedded environments, the accuracy before and after lightweight optimization remained within relatively stable intervals, and the differences were not excessively significant from a statistical perspective. The confidence intervals for inference time

Table 2: Significance analysis of experimental results (ANOVA)

Metric	Experimental Condition	F-value	p-value
Classification Accuracy (PC)	Original Model vs. Lightweight Model	5.173	0.029
Classification Accuracy (Embedded)	Original Model vs. Lightweight Model	4.651	0.039
Inference Time (PC)	Original Model vs. Lightweight Model	8.942	0.004
Inference Time (Embedded)	Original Model vs. Lightweight Model	10.331	0.002

Table 3: Confidence intervals of experimental results (95%)

Metric	Experimental Condition	Mean (%)	95% Confidence Interval (Lower - Upper) (%)
Classification Accuracy (PC)	Original Model	86.45	84.78 - 88.12
Classification Accuracy (PC)	Lightweight Model	85.23	83.59 - 86.87
Classification Accuracy (Embedded)	Original Model	82.38	80.51 - 84.25
Classification Accuracy (Embedded)	Lightweight Model	80.95	79.03 - 82.87
Inference Time (PC, ms)	Original Model	12.58	11.73 - 13.43
Inference Time (PC, ms)	Lightweight Model	9.45	8.71 - 10.19
Inference Time (Embedded, ms)	Original Model	27.32	25.96 - 28.68
Inference Time (Embedded, ms)	Lightweight Model	19.86	18.55 - 21.17

indicated that lightweight optimization improved inference speed on both PC and embedded devices, and the narrow range of confidence intervals validated the reliability of the model's optimized performance (Table 3). Overall, the lightweight model significantly reduced inference time while maintaining classification accuracy, particularly on embedded devices.

4.5 Comprehensive evaluation of model performance

In this study, the classification performance of the Logistic Regression model before and after lightweight optimization was compared by plotting ROC curves (Figure 5). The AUC values before and after optimization were 0.84 and 0.78 (on PC), and 0.83 and 0.73 (on embedded devices), respectively. As shown in the Figure 5, after lightweight optimization, the ROC curve displayed a slight decline at different false positive rates (FPR), and the AUC values decreased accordingly. This result suggests that although lightweight optimization reduced the computational complexity of the model, it also had a certain negative impact on the model's classification ability. Overall, the lightweight model still maintained a relatively reasonable performance level, particularly in the embedded environment, meeting the practical needs under resource-constrained conditions [10].

5 Discussion

The lightweight optimization significantly reduced the number of parameters and storage requirements of the Logistic Regression model, making it suitable for resource-constrained environments such as embedded devices [11]. Experimental results show that parameter pruning and 8-bit quantization greatly reduced the model's complexity, significantly decreasing the memory and computational resources required for inference, thereby improving the feasibility of embedded applications [12]. The notable improvement in inference time on both desktop computers and embedded devices

is particularly critical for embedded environments, where real-time performance is crucial for applications like waste classification with strict real-time requirements [13]. However, the lightweight optimization resulted in a decrease in classification accuracy, especially on embedded devices, where the performance loss was more pronounced. Nonetheless, the overall classification performance remained within an acceptable range [14]. The results in Table 1 and Figure 4, verified by paired t-tests, further illustrate this trade-off. Although there was a slight drop in accuracy, the statistical significance remained within a reasonable range, indicating that the lightweight model achieved a good balance between inference efficiency and performance in resource-limited embedded environments [15].

This study utilized data augmentation techniques to significantly enhance the generalization ability of the Logistic Regression model in the waste classification task [16]. Data augmentation increased the diversity of the dataset through methods such as rotation, scaling, and brightness adjustment, thereby improving the accuracy of the classification model. The p-values for these methods all showed statistical significance ($p < 0.05$), demonstrating their effectiveness [17]. In small-sample datasets, data diversity is key to preventing overfitting. With data augmentation, the model's stability in handling different feature variations was greatly improved. However, translation augmentation did not significantly improve classification performance, with a p-value of 0.071. This may be due to the fact that translation only alters the image in specific directions, failing to effectively increase overall image variability [18]. This finding suggests that different data augmentation strategies have varying effects on model performance, and future augmentation designs should be more refined, optimizing augmentation strategies based on specific tasks and data characteristics to maximize the model's generalization capability.

This study focused on analyzing the real-time performance of waste classification applications in embedded environments.

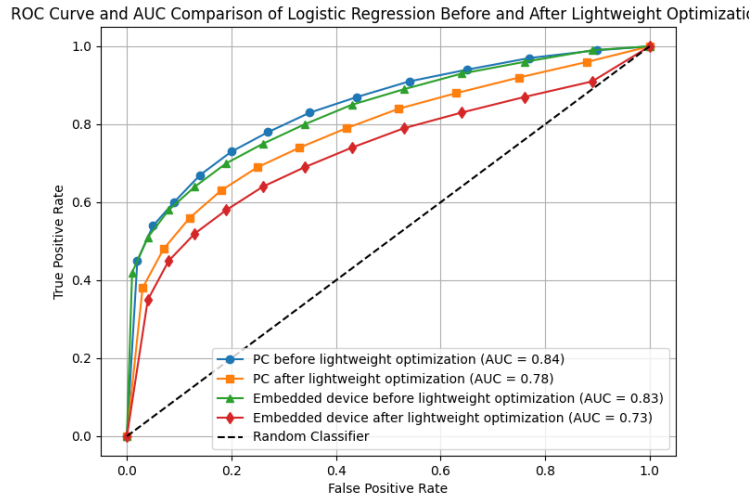


Figure 5: Comparison of ROC Curves and AUC Values Before and After Lightweight Optimization of the Logistic Regression Model.

Through task segmentation, the latency of each phase was clearly evaluated, showing that the overall system latency remained below 100 milliseconds, meeting the basic real-time requirements for embedded applications [19]. The inference process was identified as the primary bottleneck in system latency, and the lightweight optimization effectively reduced the time spent in this phase, significantly improving inference efficiency in the embedded environment [20]. Although the system as a whole met real-time performance requirements, the inference phase still accounted for more than 30% of the system's latency and remains the main target for performance improvements [21]. Future research should further optimize the inference process, combining hardware and software advancements to reduce latency, thereby enhancing the response efficiency and practical application value of embedded waste classification systems.

After lightweight optimization, the computational complexity of the Logistic Regression model was significantly reduced, but the classification performance also declined. This was particularly evident in embedded device environments, where the AUC value decreased significantly, indicating a reduction in the model's accuracy in distinguishing between different types of waste [22]. Despite this, the lightweight model demonstrated lower latency and faster response times in embedded environments, meeting the real-time requirements of IoT applications, such as waste classification tasks. Based on these results, the lightweight Logistic Regression model's low computational demands and fast response times make it an ideal choice for implementing intelligent waste classification in resource-constrained conditions. In the future, performance could be further improved by integrating more efficient feature extraction methods and optimized embedded designs.

6 Conclusion

This study successfully applied the Logistic Regression model to an embedded waste classification system through lightweight optimization and data augmentation techniques. In a resource-constrained environment, the model significantly reduced storage requirements and computational complexity through parameter pruning and quantization, while maintaining relatively high classification accuracy and low inference latency. Data augmentation further improved the model's generalization capability, allowing it to adapt to diverse waste data. Experimental results demonstrate that the lightweight optimization achieved good real-time performance in embedded environments, verifying the feasibility and application potential of this approach in IoT waste classification systems. Future research may explore more efficient feature extraction and embedded optimization methods to further enhance system performance.

References

- [1] Shrivani, N., & Vigneshwari, S. (2023). Optimizing Waste Management through Intelligent Machine Learning Systems: A Comparative Study. 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1-6. <https://doi.org/10.1109/ICCCNT56998.2023.10308278>
- [2] Uganya, G., Rajalakshmi, D., Teekaraman, Y., Kuppusamy, R., & Radhakrishnan, A. (2022). A Novel Strategy for Waste Prediction Using Machine Learning Algorithm with IoT Based Intelligent Waste Management System. *Wireless Communications and Mobile Computing*, (1): 2063372. <https://doi.org/10.1155/2022/2063372>
- [3] Mouhajir, M., Nechba, M., & Sedjari, Y. High Performance Computing Applied to Logistic Regression: A CPU and GPU Implementation Comparison. 2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings). IEEE, 2023: 1-5. <https://doi.org/10.1109/AIBThings58340.2023.10291024>
- [4] Nnamoko, N., Barrowclough, J., & Procter, J. (2022). Solid Waste Image Classification Using Deep Convolutional Neural Network, *Infrastructures*, 7(4): 47. <https://doi.org/10.3390/infrastructures7040047>
- [5] Lupei, M., Li, D., Ingraham, N., Baum, K., Benson, B., et al. (2022). A 12-hospital Prospective Evaluation of a Clinical Decision Support Prognostic Algorithm Based on Logistic Regression as a Form of Machine Learning to Facilitate Decision Making for Patients with Suspected COVID-19. *PLoS ONE*, 17(1): e0262193.

- <https://doi.org/10.1371/journal.pone.0262193>
- [6] Solovyeva, E., & Abdullah, A. (2022). Comparison of Different Machine Learning Approaches to Text Classification. 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 1427-1430. <https://doi.org/10.1109/ElConRus54750.2022.9755806>
 - [7] Yadav, A., Kulpriya, S., & Upadhyay, D. (2023). A Statistical Analysis for Heart Disease Prediction System for Next-Gen Software. 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), 71-76. <https://doi.org/10.1109/cictn57981.2023.10140436>
 - [8] Lou, L., & Gou, N. (2023). Domestic Waste Image Classification Algorithm Based on Improved EfficientNetV2. 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP), 1374-1377. <https://doi.org/10.1109/ICSP58490.2023.10248442>
 - [9] Li, T., Li, X., & Li, C. (2022). Garbage Classification Algorithm Based on Improved Lightweight Network ShufflenetV2. 2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT), 1-4. <https://doi.org/10.1109/aicit55386.2022.9930200>
 - [10] Loganayagi, S., & Usha, D. (2023). An Automated Approach to Waste Classification Using Deep Learning. 2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT), 01-10. <https://doi.org/10.1109/ICECCT56650.2023.10179743>
 - [11] Xiao, Z., Yang, G., & Wang, X. (2023). Lightweight Garbage Detection Algorithm Based on Improved YOLOv5s. Proceedings of the 2023 7th International Conference on Big Data and Internet of Things, 82-92. <https://doi.org/10.1145/3617695.3617710>
 - [12] Huang, M., Chang, Y., Zhang, L., & Jiao, S. (2022). A Lightweight YOLOv5 Garbage Detection and Classification Method. Proceedings of SPIE, 12473: 203-210. <https://doi.org/10.1117/12.2653673>
 - [13] Zhao, Y., Huang, H., Li, Z., Huang, Y., & Lu, M. (2022). Intelligent Garbage Classification System Based on improve MobileNetV3-Large. Connection Science, 34(1): 1299-1321. <https://doi.org/10.1080/09540091.2022.2067127>
 - [14] Cai, X., Shuang, F., Sun, X., Duan, Y., & Cheng, G. (2022). Towards Lightweight Neural Networks for Garbage Object Detection. Sensors, 22(19): 7455. <https://doi.org/10.3390/s22197455>
 - [15] Liu, J., Cui, J., & Chen, C. (2023). Online Efficient Secure Logistic Regression Based on Function Secret Sharing. Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 1597-1606. <https://doi.org/10.1145/3583780.3614998>
 - [16] Rutqvist, D., Kleyko, D., & Blomstedt, F. (2020). An Automated Machine Learning Approach for Smart Waste Management Systems. IEEE Transactions on Industrial Informatics, 16(1): 384-392. <https://doi.org/10.1109/TII.2019.2915572>
 - [17] Gomez, D. O., Toro, M., & Morales, W. (2021). Solid Domestic Waste Classification Using Image Processing and Machine Learning. OSF Preprints. <https://doi.org/10.31219/osf.io/yzcfk>
 - [18] Hussain, A., Draz, U., Ali, T., Tariq, S., Irfan, M., et al. (2020). Waste Management and Prediction of Air Pollutants Using IoT and Machine Learning Approach. Energies, 13(15): 3930. <https://doi.org/10.3390/en13153930>
 - [19] Li, J., Haq, A., Din, S. U., Khan, J., Khan, A., & Saboor, A. (2020). Heart Disease Identification Method Using Machine Learning Classification in E-healthcare. IEEE Access, 8: 107562-107582. <https://doi.org/10.1109/ACCESS.2020.3001149>
 - [20] Ahmad, K., Khan, K., & Al-Fuqaha, A. (2020). Intelligent Fusion of Deep Features for Improved Waste Classification. IEEE Access, 8, 96495-96504. <https://doi.org/10.1109/ACCESS.2020.2995681>
 - [21] Shi, C., Tan, C., Wang, T., & Wang, L. (2021). A Waste Classification Method Based on a Multilayer Hybrid Convolution Neural Network. Applied Sciences, 11(18): 8572. <https://doi.org/10.3390/app11188572>
 - [22] Ghavamipour, A. R., Turkmen, F., & Jian, X. (2021). Privacy-preserving Logistic Regression with Secret Sharing. BMC Medical Informatics and Decision Making, 22(1): 89. <https://doi.org/10.1186/s12911-022-01811-y>