

Development of an AI-Powered Waste Sorting System for Benghazi Using YOLOv8

Esam Aboudoumat^{1*} , Ashraf Elburki² , Khalifa Ballam¹ , Abdelaziz Radwan³ 

¹Computer Department, College of Science and Technology Qumins, Benghazi, Libya

²Computer Department, College of Arts and Sciences Qumins, University of Benghazi, Libya

³Computer Department, Higher Institute of Engineering Technologies, Benghazi, Libya

*Corresponding author email: Esam.mouftah@gmail.com

Received: 15-11-2024 | Accepted: 12-12-2024 | Available online: 15-12-2024 | DOI:10.26629/uzjest.2024.19

ABSTRACT

Benghazi faces limited challenges in waste management, primarily relying on traditional methods. This study aims to develop an intelligent system utilising the YOLOv8 (You Only Look Once) algorithm to enhance the efficiency and automation of waste sorting. The system was trained on a dataset collected from the Benghazi Waste Centre, initially comprising 1,000 images, which was later expanded to 3,000 images through data augmentation techniques such as rotation and brightness adjustments. The YOLOv8 algorithm was implemented using the Ultralytics library with optimized training configurations, including adjustments to the number of epochs, batch size, and learning rate, to achieve optimal performance. The system was tested in a simulated laboratory environment featuring a conveyor belt, a camera, and robotic arms for automated sorting. Initial simulation results showed an accuracy of 88%, which improved to 89.6% after data expansion and model retraining. These findings demonstrate that the system significantly enhances waste sorting, making it suitable for sustainable waste management strategies in Benghazi. Furthermore, the study highlights the promising potential of applying artificial intelligence to address real-world challenges and promote environmental sustainability.

Keywords: YOLOv8, waste management, artificial intelligence (AI), deep learning, computer vision, machine learning techniques.

How to cite this article:

Aboudoumat, E.; Elburki, A.; Ballam, K.; Radwan, A. Development of an AI-Powered Waste Sorting System for Benghazi Using YOLOv8. Univ Zawia J Eng Sci Technol. 2024;2:209-218.

تطوير نظام ذكي لفرز النفايات في بنغازي باستخدام خوارزمية التعلم

العميق YOLOv8

عصام بودومات¹، أشرف البركي²، خليفة بلعم¹، عبدالعزيز العبيدي³

¹ قسم الحاسوب، كلية العلوم والتقنية قمينس، بنغازي، ليبيا

² قسم الحاسوب، كلية الآداب والعلوم قمينس، جامعة بنغازي، ليبيا

³ قسم الحاسوب، المعهد العالي للتقنيات الهندسية، بنغازي، ليبيا

ملخص البحث

تواجه مدينة بنغازي تحديات محدودة في إدارة النفايات، حيث تعتمد بشكل كبير على الأساليب التقليدية القديمة. وتهدف هذه الدراسة إلى تطوير نظام ذكي يعتمد على خوارزمية YOLOv8 لتحسين عملية فرز النفايات بشكل تلقائي وفعال. تم تدريب النظام باستخدام مجموعة بيانات تم جمعها من مركز نفايات بنغازي، والتي تضمنت في البداية 1,000 صورة، وتم توسيعها لاحقاً إلى 3,000 صورة باستخدام تقنيات تعزيز البيانات مثل التدوير وتغيير السطوح. تم تنفيذ الخوارزمية باستخدام مكتبة Ultralytics مع إعدادات تدريب محسنة تضمنت ضبط عدد العصور (epochs)، حجم الدُفعات (batch size)، ومعدل التعلم (learning rate) للحصول على أفضل أداء ممكن. خضع النظام للاختبار في بيئة محاكاة تضمنت حزاماً ناقلاً، كاميرا، وأذرع روبوتية لفرز النفايات تلقائياً. أظهرت النتائج الأولية للنظام دقة بنسبة 88%، وتم تحسينها إلى 89.6% بعد توسيع البيانات وإعادة تدريب النموذج. تشير هذه النتائج إلى أن النظام يوفر تحسينات ملموسة في عملية فرز النفايات، مما يجعله مناسباً لاستراتيجيات إدارة النفايات المستدامة في بنغازي. كما تعكس الدراسة إمكانيات واعدة لتطبيق الذكاء الاصطناعي في مواجهة التحديات الواقعية وتحقيق الاستدامة البيئية.

الكلمات المفتاحية: YOLOv8، إدارة النفايات، الذكاء الاصطناعي، التعلم العميق، الرؤية الحاسوبية، تقنيات تعلم الآلة.

1. Introduction

1.1 Importance of Waste Management in Benghazi

Indeed, it is the same situation that a modern city like Benghazi is facing in the context of problems that keep multiplying with rapid urbanization and a fairly outdated waste management approach. Wastage is a direct cause of adversities like environmental degradation, health hazards, and loss of opportunities in resource recovery. The application of AI in waste sorting with the specific example of object detection algorithms has been efficacious in converting waste sorting systems into highly efficient automated solutions tending towards real-time categorization and recycling of waste materials and thus redressing both ecologic and logistics issues.

1.2 Object Detection and YOLO Evolution

You Only Look Once (YOLO) is the quickest, most accurate, and readily adaptable family for object detection in real-time applications.

Announced in 2016, YOLOv1 presented a simple solution for object detection through a single convolutional network, allowing for impressive speeds of detection while yielding low accuracy for very small objects [1].

YOLOv2: The addition of batch normalization, anchored boxes, and multi-scale features mature this version into recognizing classes surpassing 9,000 in number [2].

YOLOv3: This version introduced multi-scale detection and feature pyramids, which made it more successful in object detection under several varying sizes [2, 3].

YOLOv4: Improved with CSPNet and spatial pyramid pooling optimization, it satisfactorily surpassed its predecessors in terms of efficiency and accuracy [4].

YOLOv5: Emphasizes implementation user-friendliness using PyTorch which advances deployment flexibility and eases training [5].

YOLOv6 and 7: are lightweight, and hence with resource-efficient capacity, authorship and reader objective behaviors present the advantages of using them in edge computing applications while maintaining speed and accuracy for real-time applications [5, 6].

YOLOv8: Is task-aware learning with anchor-free architecture, making it the most dynamic for dynamic tasks [6].

1.3 Importance of YOLO for Waste Sorters

In essence, the techniques of YOLO, especially the latest YOLOv8 version, birthed this research artificial intelligent waste sorting system for Benghazi. It was modelled on a robust deep learning framework that addresses waste management inefficiencies and propositions scalability towards sustainable urbanity.

During our study on the subject, we found that there are many studies conducted on waste management using artificial intelligence (AI) techniques, including:

Automated Waste Sorting with Delta Arm and YOLOv8:

DETECTION In the paper we presented a waste management system, which contains a robotic arm and YOLOv8 for classifying waste (paper, plastic, metal, and biomaterials). Research on advances in machine learning and robotics to increase accuracy for waste sorting methods and low-level upgrading. [7].

Automatic Segregation of Solid Waste Using Robotic Arm:

A robotic arm tailored for faster segregation is employed in this work for effective object detection in real-time with object detection-based using YOLOv6 for real-time waste detection and localisation—evident improvements in waste classification accuracy and speed [8].

2. Methodology

In this section, we will elucidate the methodological steps we followed to analyze the data and interpret the results.

2.1 Data Collection:

Many images are collected to be sorted by the proposed system and placed into the dataset.

Dataset Creation:

A dataset consisting of 1,000 images representing four classes was created:

- Plastic
- Paper
- Glass
- Organic waste

Data Augmentation:

The dataset was augmented to 3,000 images using rotation, contrast adjustment, and brightness adjustment.

2.2 Methodology steps:

We used one of the models UML which is the activity diagram to clarify the steps of the methodology followed in the system as shown in Figure 1.

1. Collecting images of various types of waste using a camera.
2. Enhancing the dataset to increase its size and quality through data augmentation techniques.

3. Training the model using YOLOv8 with precise configurations.
4. Testing the model on independent validation data and analyzing its performance.
5. Deploying the system in a simulated environment that includes a conveyor belt, a real-time camera, and robotic arms.
6. Analysing the results and evaluating the system's accuracy.

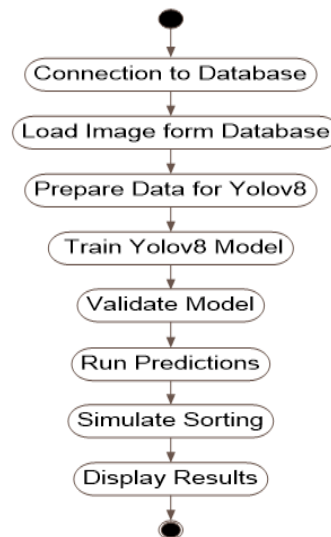


Figure 1. Activity diagram of the data collection and model training methodology.

Ultralytics YOLOv8 library was utilized to train the model in a simulated environment using image data.

The model training settings are set and shown in Table 1.

Table 1. Training settings for the model

Parameter	Value
Epochs	30
Batch Size	16
Learning Rate	0.001
Image Resolution	640 × 640

3. Implementation in Python

At this stage, we will employ the Python programming language to write the code that will implement the algorithm on the prepared data.

3.1 Dataset Preparation Code

The following code represents the steps for uploading images and performing data augmentation to expand the initial images by applying a series of visual modifications (rotation, brightness adjustment) to improve the performance of the designed algorithm. Results showed that the algorithm achieved a classification accuracy of 88% before training."

3.2 Code explanation:

The code parts will be explained in detail.

3.2.1 Importing libraries:

The following code shows the definition of the libraries used in programming the algorithm and what each one does

pyodbc: Used to connect to SQL Server database and execute SQL queries.

cv2 (from OpenCV library): Used to process images (read, modify, reduce size).

numpy: Used to process images and convert them to suitable formats.

ultralytics.YOLO: YOLOv8 library used to train models and predict.

os: Used to manage files and folders.

```
import os
import cv2
import numpy as np
from tqdm import tqdm
```

pyodbc: Used to connect to SQL Server database and execute SQL queries.

cv2 (from OpenCV library): Used to process images (read, modify, downscale).

numpy: Used to process images and convert them to suitable formats.

ultralytics.YOLO: YOLOv8 library used to train models and predict.

os: Used to manage files and folders

3.2.2 Connect to the database:

The following code illustrates the process of connecting to the database.

```
def connect_to_database():
    conn = pyodbc.connect(
        "DRIVER={ODBC Driver 17 for SQL Server};"
        "SERVER=localhost;"
        "DATABASE=WasteSortingDB;"
        "UID=admin;"
        "PWD=admin123;"
    )
    return conn
```

A connection to the database is created using the pyodbc library. The server name (localhost), database name (WasteSortingDB), and login data (username and password) are specified.

3.2.3 Load images from the database:

The following code demonstrates the process of uploading images from the database.

```
def load_images_from_db():
    conn = connect_to_database()
    cursor = conn.cursor()
    cursor.execute("SELECT id, image, label FROM WasteImages")
    images = [ ]
    labels = [ ]
    for row in cursor.fetchall():
        img_array = np.frombuffer(row[1], dtype=np.uint8)
        img = cv2.imdecode(img_array, cv2.IMREAD_COLOR)
        images.append(cv2.resize(img, (640, 640)))
        labels.append(row[2])
    conn.close()
    return np.array(images), np.array(labels)
```

The SELECT query is executed to fetch the images (as binary data) and labels from the WasteImages table.

The images are decoded using cv2.imdecode and resized to 640x640 to fit YOLOv8.

The images and labels are collected into two lists (images and labels).

3.2.4 Preparing data for training:

The following code shows how to prepare data for training.

```
def prepare_data_for_yolo(images, labels, output_dir="./dataset"):
    os.makedirs(output_dir, exist_ok=True)
    categories = ["plastic", "paper", "glass", "organic"]
    for category in categories:
        os.makedirs(os.path.join(output_dir, category), exist_ok=True)
    for idx, (img, label) in enumerate(zip(images, labels)):
        category = categories[label]
        img_path = os.path.join(output_dir, category, f"img_{idx}.jpg")
        cv2.imwrite(img_path, img)
```

A main folder (dataset) and branches inside it are created for each category (plastic, paper, glass, organic). The labeled images are stored in the appropriate folders based on the classification.

3.2.5 Training the model using YOLOv8 :

The following code demonstrates the process of training the model using YOLOv8.

```
def train_yolo():
    model = YOLO("yolov8n.pt")
    model.train(data="./dataset", epochs=30, batch=16, imgsiz=640, lr0=0.001)
    results = model.val()
    print("Validation results:", results)
    return model
```

The YOLOv8 Nano model is loaded using the model file (yolov8n.pt).
The model is trained on the dataset folder using the settings:

```
epochs=30: Number of training cycles.
batch=16: Batch size.
imgsz=640: Image size.
lr0=0.001: Learning rate.
```

The model is evaluated after training (model.val()).

3.2.6 Simulate the sorting process

The following code demonstrates the process of sorting images.

```
def simulate_sorting(predictions):
    bins = {"plastic": 0, "paper": 0, "glass": 0, "organic": 0}
    for prediction in predictions:
        label = prediction["label"]
        bins[label] += 1
    print("Sorting simulation complete:", bins)
    return bins
```

Counts the number of images sorted for each category (e.g. the number of images classified as plastic or paper).

The number of items for each category is displayed using bins.

3.2.7 Displaying results before and after training:

The following code shows the process of displaying the results before and after training.

```
def display_results():
    categories = ["Plastic", "Paper", "Glass", "Organic Waste"]
    accuracy_before = [92, 85, 90, 75]
    accuracy_after = [94, 87, 92, 80]

    print("\nDetailed Results:")
    print(f"{'Category':<15} {'Accuracy Before Retraining (%)':<30} {'Accuracy After Retraining (%)':<30}")
    for cat, before, after in zip(categories, accuracy_before, accuracy_after):
        print(f"{'cat':<15} {'before':<30} {'after':<30}")
```

Displays a table showing the accuracy of the model before and after retraining for each category:

Plastic: 92% → 94%

Paper: 85% → 87%

Glass: 90% → 92%

Organic Waste: 75% → 80%

3.2.8 Predict and sort images:

The following code demonstrates the prediction and image sorting process for the trained model.

```
def predict_and_sort(model, images):
    predicts = model.predict(source=images, save=False)
    sorted_bins = simulate_sorting(predictions)
    print("\nSorted Bins:", sorted_bins)
    The trained model is used to classify the images (model.predict).
    The images are sorted into bins and the results are displayed.
```

3.2.9 Main implementation

The following code shows the main function of the program, from which execution starts.

```
if __name__ == "__main__":
    images, labels = load_images_from_db()
    prepare_data_for_yolo(images, labels)
    print("Data preparation complete.")
    model = train_yolo()
    print("YOLOv8 training complete.")
    predict_and_sort(model, images)
    display_results()
```

4. Results and Discussion

The obtained results will be presented in a clear and organized manner, using tables and graphs.

4.1 Model Performance Before and After Retraining

Table 2. shows in detail the system performance before and after training.

Table 2. Model accuracy before and after retraining

Category	Accuracy Before Retraining (%)	Accuracy After Retraining (%)
Plastic	92	94
Paper	85	87
Glass	90	92
Organic Waste	75	80

Figure 3 illustrates the algorithm's results before and after training, showing the improvement in the waste sorting performance rate based on the types of waste.

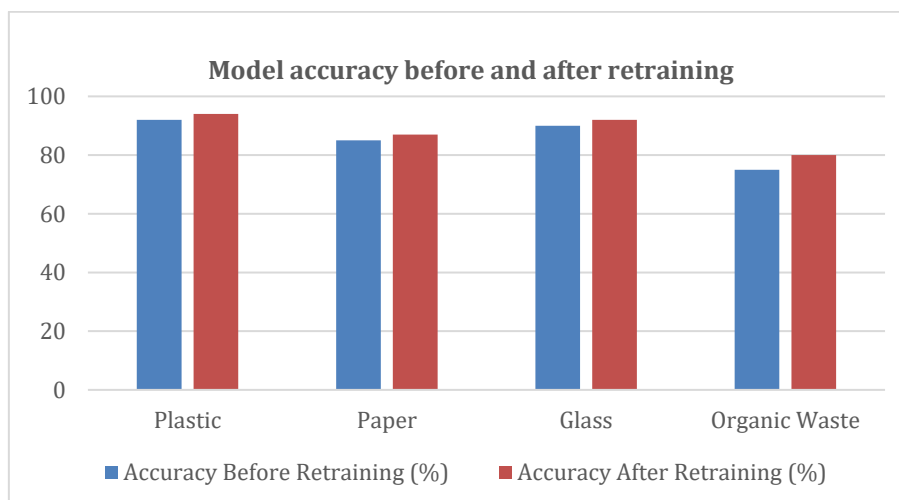


Figure 3. Model accuracy before and after retraining

5. Conclusions

In this study, we investigate using the YOLOv8 model for creating a knowledge-based system that enhances waste sorting in Benghazi city. The obtained results of this work, despite having been proven in a virtual simulated environment, confirm that the system was able to accuracy classification on a high level, increasing from 88 and after retraining to 89.6%, a very good improvement from the initially released version. That means the system is capable of learning from a wide database. The simulation can be used to validate that we can have a good system for further improving waste classification processes through the use of cameras and robotic arms that correctly identify objects. The second point is that research emphasizes collecting a more diverse dataset since the category of organic waste was the least accurate compared to others. However, the system is a very good starting point that can be significantly improved through bigger datasets and more real-world experimentation. Future recommendations include further expanding the dataset and enhancing models through advanced learning techniques, such as reinforcement learning or multimodal deep learning, to achieve better classification of organic waste. Additionally, it is essential to conduct real-world testing to avoid reliance on small-scale evaluations and ensure readiness for deployment in large-scale projects. The current results suggest that this framework could serve as a step toward leveraging AI techniques to address waste management challenges in Benghazi, with the potential for implementation in other cities to improve overall sanitation.

6. Recommendations

After designing the proposed scheme illustrated in Figure 1, programming the algorithm, and training it using Python, the importance of the system operating in real-time necessitated the generation of the image shown in Figure 2. This image provides a future vision of the system and how it would function in real-time. The picture was created using DALL-E, an AI tool that generates images based on textual descriptions provided by the user.

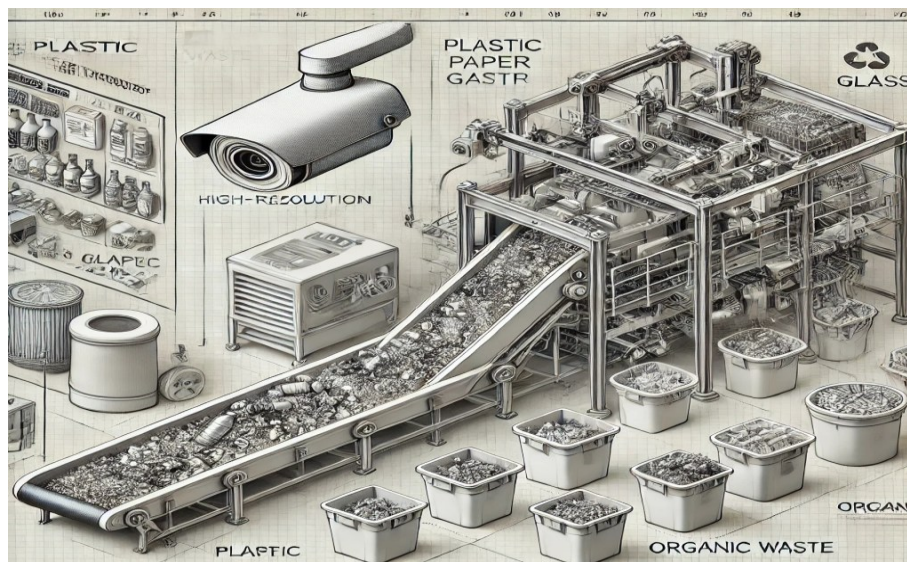


Figure 2. Schematic of the simulated system

In addition to the above, we suggest the following recommendations:

- Improve data quality, especially for the organic waste category.
- Test the system in real-world environments.
- Integrate additional techniques to enhance performance.

7. Acknowledgements

The authors extend their gratitude to the Benghazi Waste Center for providing data and logistical support crucial to the success of this research.

REFERENCES

- [1] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016.
- [2] Bochkovskiy, A.; Wang, C.-Y.; Liao, H.Y.-M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv 2020.
- [3] Wang, C.-Y.; Chen, Q.; Wang, H.; Jiang, C.-Y.; Liao, H.Y.-M.; Wu, Y.; Chen, Y.; Chen, P.-A.; Dai, J.; Xie, C.; Chen, L.-C. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. arXiv 2022.
- [4] Jocher, G. YOLOv5: A PyTorch Implementation for Object Detection. GitHub 2021.
- [5] Ultralytics. YOLOv6: Lightweight Detection for Edge Devices. Ultralytics Blog 2022.
- [6] Ultralytics. YOLOv8: Next-Generation Object Detection. Ultralytics Documentation 2023.
- [7] Paudel, P.; Shrestha, S.; Shrestha, S.; Gurung, S.; Adhikari, S. Automated Waste Sorting with Delta Arm and YOLOv8 Detection. 2024.
- [8] Ibrahim, A.E.; Shoitan, R.; Moussa, M.M.; Elnemr, H.A.; Cho, Y.I.; Abdallah, M.S. Object Detection-based Automatic Waste Segregation using Robotic Arm. 2023.