DS Journal of Artificial Intelligence and Robotics
Volume 3 Issue 2, 1-25, Apr - Jun 2025
ISSN: 2583-9926 / https://doi.org/10.59232/AIR-V3I2P101

Original Article

Combining YOLO and Scikit-Learn Improves Real-World Audience Classification

Tri Luu[1], Thinh Tong[1], Tuong Dang[1], Vuong Pham[1], Minh Phan[1]*

[1]*Sai Gon University, Ho Chi Minh City, Vietnam.*

*minhpn@sgu.edu.vn

Received:02 April 2025; Revised: 04 May 2025; Accepted: 06 June 2025; Published: 30 June 2025;

Abstract - This study addresses the problem of automatic object classification by leveraging the strengths of both deep learning and traditional machine learning. The main goal of this project is to develop a prototype application capable of efficiently and accurately recognizing and classifying objects in images. To tackle this, the YOLOv10 model for object detection was used, then extracted features such as bounding-box size [3] and average color. If an image is of poor quality or YOLOv10 fails to detect any object, this study applies PCA to enhance image quality. These extracted features are then used to train a Random Forest classifier from the scikit-learn library. The performance of the Random Forest classifier is optimized using GridSearchCV [2] and evaluated using StratifiedKFold [5]. The results showed that the YOLO + Random Forest combination system achieved an overall accuracy of 93%, with a higher average Precision and F1-score than using YOLOv10 alone. The combined model significantly improves the ability to classify glass and organic objects, although the number of samples of these two types is limited. The study concluded that the combination of YOLOv10 and Random Forest is an effective approach to building an automated object classification system, taking advantage of the detection speed of deep learning and the characterization-based classification capabilities of traditional machine learning, contributing to intelligent object management.

Keywords - Feature extraction, Image processing, Object classification, Random forest, YOLOv10.

1. Introduction

Automatic object classification systems play an increasingly important role in industrial automation, waste management, and intelligent environmental monitoring. Although deep learning models, such as YOLO, have significantly advanced the capabilities of object detection, these approaches still face challenges in accurately classifying objects under difficult conditions such as low-light environments, reflective surfaces, and visually similar items. Meanwhile, traditional machine learning methods, known for their robust feature-based classification abilities, have not been extensively integrated with modern deep-learning frameworks.

This study proposes a novel hybrid approach that integrates YOLOv10—an advanced deep learning model—with the Random Forest classifier from the Scikit-learn library. Principal Component Analysis (PCA) is utilized as a supplementary step for image preprocessing to enhance feature visibility in challenging imaging scenarios. The main goal is to combine YOLOv10's real-time detection strengths with Random Forest's feature-based classification capabilities, aiming to significantly improve classification accuracy in practical applications such as automated recycling and waste sorting.

The proposed methodology was thoroughly evaluated on the Trash Detection dataset from Roboflow, which contains six challenging object categories: cardboard, glass, metal, organic, paper, and plastic. The experimental

evaluation demonstrates how the integrated YOLOv10, Random Forest, and PCA pipeline effectively addresses the shortcomings of standalone YOLO, especially in cases of visually challenging objects, resulting in improved accuracy and reliability.

2. Review of Literature

2.1. YOLOv10 Model

YOLO (You Only Look Once) is one of the most popular object detection architectures known for its fast and accurate real-time detection capabilities. YOLOv10 is the latest version, building upon improvements from previous versions such as YOLOv4, YOLOv5, and YOLOv7, with optimized network architecture and training algorithms.

The core principle of YOLO is dividing the input image into grid cells, where each cell predicts bounding boxes and class probabilities for objects within that region. YOLOv10 features enhancements like a lighter backbone, feature pyramid networks for multi-scale detection, and improved training methods to boost accuracy and speed.

Advantages of YOLOv10 include:

- High-speed object detection suitable for real-time applications.
- Ability to detect multiple object classes with good accuracy.
- Integration of advanced data augmentation and training techniques.

However, YOLOv10 can struggle with visually similar objects and may have reduced performance under challenging conditions such as low light or blurry images.

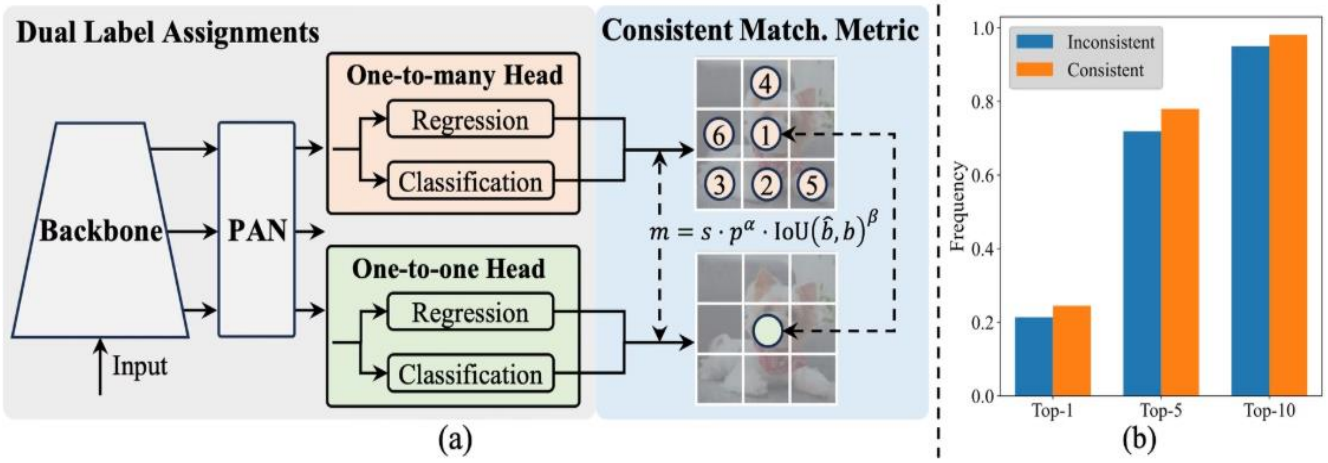


Fig. 1(a) Consistent dual assignments for NMS-free training in YOLOv10, and (b) Frequency of one-to-one assignments in top-1/5/10 of one-to-many results for YOLOv8-S which employs α_{o2m}=0.5 and β_{o2m}=6 by default. For consistency, α_{o2o}=0.5, β_{o2o}=6. For consistency, α_{o2o}=0.5, β_{o2o}=6.

2.2. Random Forest Algorithm

Random Forest (RF) is a machine learning algorithm based on constructing an ensemble of decision trees built on randomly selected subsets of data and features. Each tree gives a classification vote, and the final prediction is made by majority voting.

RF works by:

- Building many decision trees from bootstrapped samples of the training data.
- Training each tree on a random subset of features.
- Combining predictions from all trees to reduce overfitting and improve generalization.

Its advantages include:

- Strong classification performance on high-dimensional data without strict assumptions about data distribution.
- Robustness to noise and outliers.
- Ease of tuning and applicability too small to medium datasets.

The downsides include potentially slower prediction times with many trees and reduced interpretability compared to simpler models.

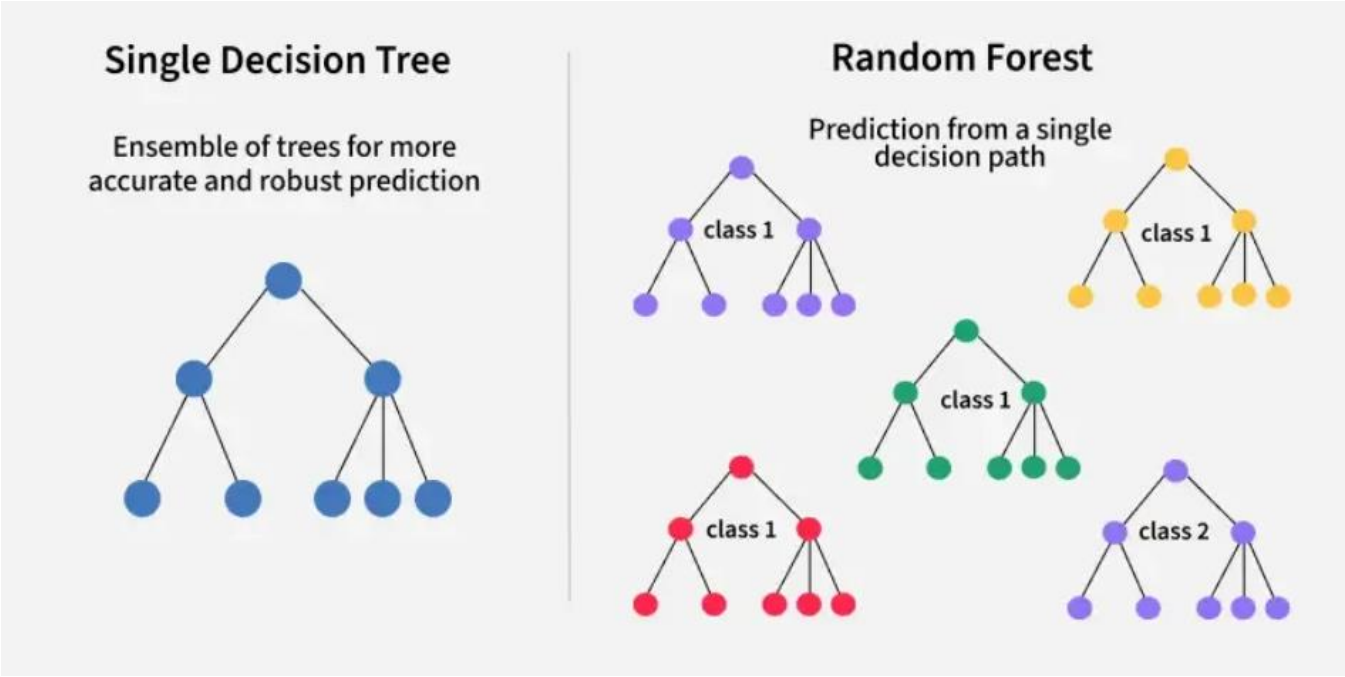


Fig. 2 Comparison between a single decision tree and a random forest ensemble

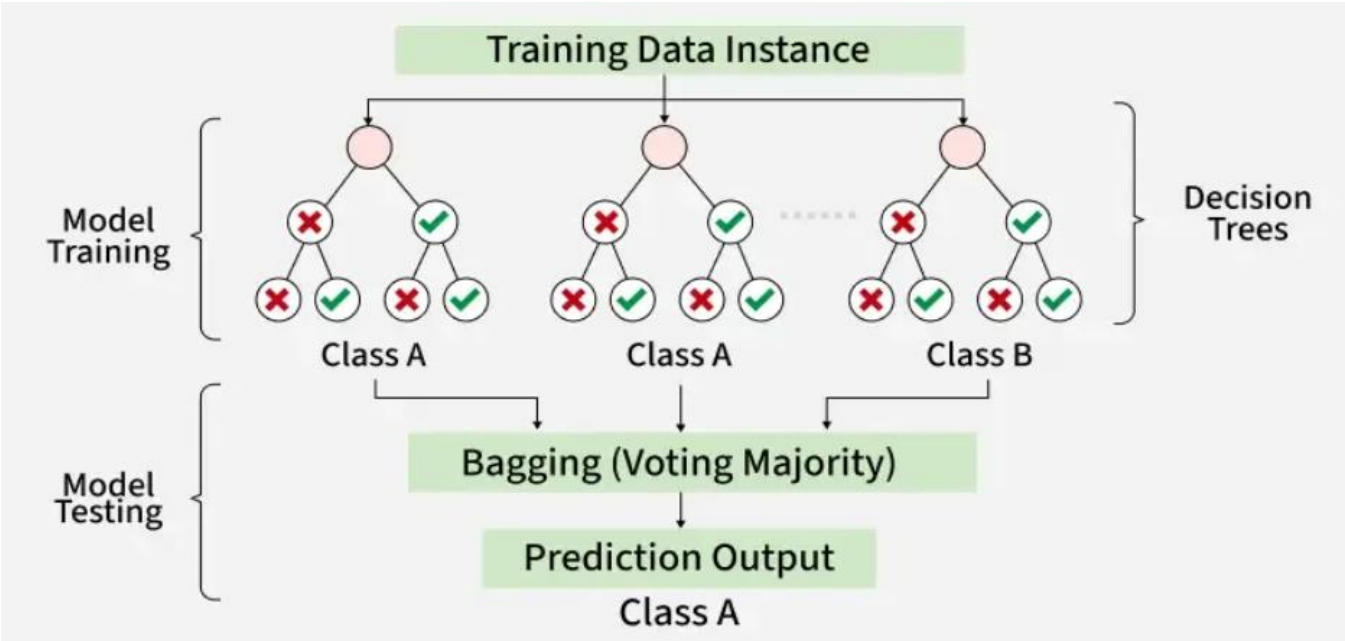


Fig. 3 Random forest training and testing workflow with bagging and majority voting

2.3. Principal Component Analysis (PCA)

Principal Component Analysis is a dimensionality reduction technique that transforms original data into a new coordinate system defined by principal components, which capture the directions of maximum variance.

In image processing, PCA is used to:

- Remove noise by filtering out less important components.
- Enhance image quality, especially for blurred or poorly lit images, by reconstructing images from principal components.

PCA operates by:

- Representing each pixel as a vector in color space.
- Finding new orthogonal axes that capture the most variance.
- Keeping only a few principal components to reduce dimensionality.
- Reconstructing images from these components to smooth and denoise while preserving important structures.

Applying PCA in this study helps improve input image quality, thereby aiding YOLOv10 in better detecting challenging objects.

2.4. Justification for Model Combination

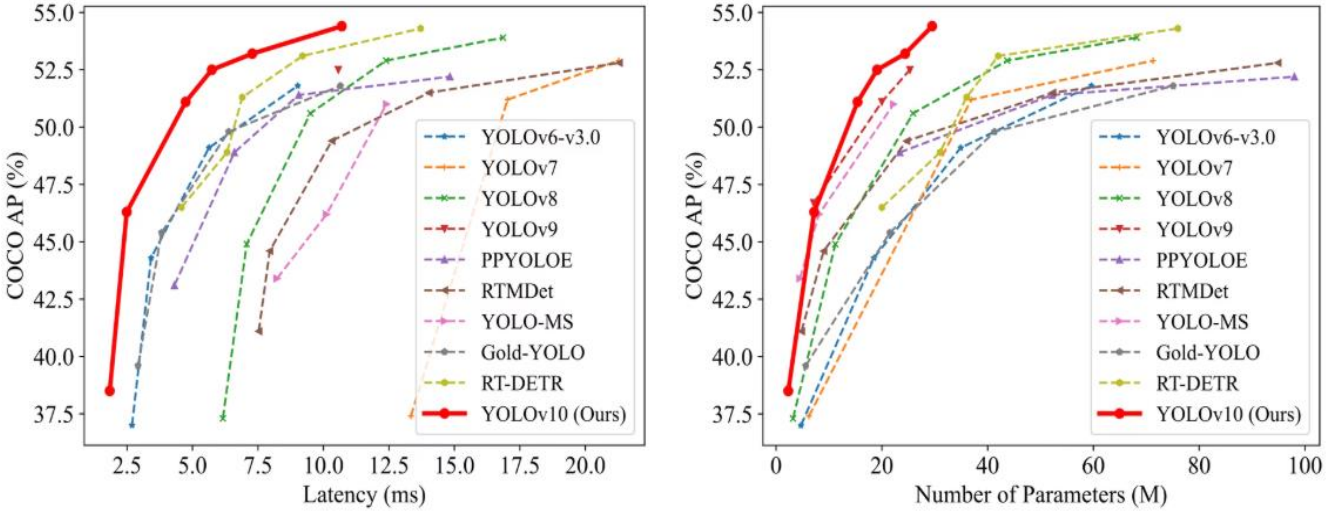


Fig. 4 Comparison of YOLOv10 with other modern object detection models on the COCO dataset

YOLOv10 serves as the primary detection module due to its advanced architecture, which delivers improved accuracy and lower computational cost compared to earlier versions such as YOLOv5 or YOLOv8. However, despite its strong performance, YOLOv10 can still struggle with detecting objects under poor lighting conditions, occlusion, or cluttered backgrounds. To address this limitation, Principal Component Analysis (PCA) is applied as a preprocessing step to enhance image clarity and amplify key visual cues. This helps improve YOLOv10's object detection capabilities in degraded visual environments.

Following the detection phase, additional features are extracted from the detected bounding boxes. These include geometric properties (e.g., box dimensions, aspect ratios), statistical color values (e.g., average RGB), and deep features obtained from intermediate layers of the YOLO model. These combined features provide a richer representation of the object beyond standard detection outputs. A Random Forest classifier is then used to perform a secondary classification step, particularly in cases where the initial detection is uncertain or involves visually

similar categories. This layered design enhances classification reliability without introducing significant computational complexity.

3. Method

3.1. Research Methods

In this project, the research team developed a synchronous research-experiment process, starting with an in-depth survey of the YOLOv10 object detection algorithm and popular machine learning methods in the Scikit-learn library to solve the classification problem. In terms of data preparation, the team collected thousands of images from a variety of sources—including actual shots taken at living areas and waste treatment facilities—and then performed pre-processing steps such as size balancing, brightness-contrast adjustment, and noise removal to ensure the consistency and richness of the data set. Next, the YOLOv10 model is fine-tuned on this specialized dataset, ensuring the accurate detection of the bounding boxes of objects with optimal confidence thresholds. From the detection results, the system automatically extracts "rough" features such as area and aspect ratio and analyzes the color histogram in the HSV space to describe the color information. In cases where the image is blurry or contains a lot of noise—such as shadows or low-light conditions—the PCA (Principal Component Analysis) [1] method is applied to filter out random components, thereby shortening the data on the main components and improving the quality of the input signal.

The second classification layer uses the Random Forest algorithm, with hyperparameters optimized via GridSearchCV combined with StratifiedKFold to ensure generalization and avoid overfitting. Evaluation metrics [11], including Accuracy, Precision, Recall, and F1-score, are calculated for each class and macro mean, thereby allowing direct comparison of performance between the native YOLOv10 model, pure Random Forest, and the combined method. Finally, the study proposes a "fusion" mechanism between the two prediction streams: prioritizing the output of YOLOv10 when the confidence is high, and switching to using labels from Random Forest in cases of low confidence or when YOLO fails to detect the bounding box [3]. This mechanism exploits the speed and sensitivity advantages of YOLOv10 and the sophisticated discrimination of Random Forest, resulting in overall classification results that are superior to individual methods.

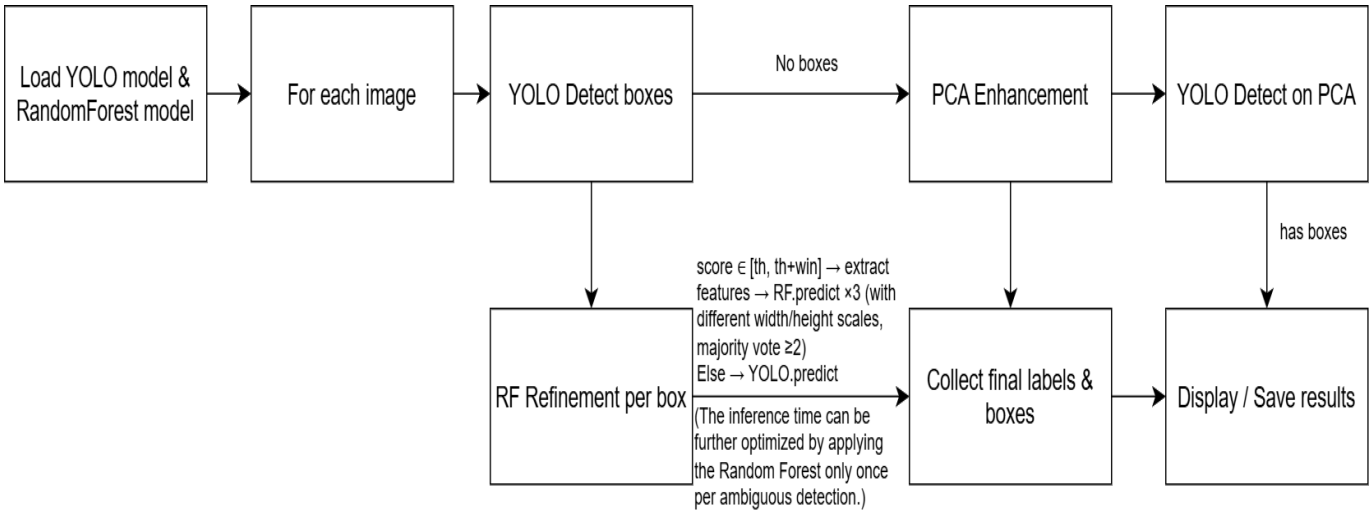


Fig. 5 YOLO + Random Forest + PCA system stream processing

3.2. System Operation Process

3.2.1. Feature Extraction from Images

The main purpose of this procedure is to use the YOLOv10 model trained in the datasets file that is being trained for Random Forest to extract the deep features [13] required for training the Random Forest classifier.

The steps include:

- Running the YOLOv10 model on each image retains only the findings (it is possible to set the reliability beyond the preset threshold (e.g. 0.35) so that the desired data can be limited).
- From the bounding boxes detected, the usual features such as size (width, height) and average color (channels B, G, R) of the image area containing the object can be extracted. In contrast, the deep features can be obtained from the BACKBONE layer of the yolo model.
- Store these features along with the object's label (taken from YOLO's prediction results) and information about the data source (e.g., "train") into a CSV file.
- This process creates a dataset of extracted features, ready to be used in training the Random Forest classifier.

3.2.2. Training the Random Forest classifier

The main purpose is to train the Random Forest classifier to classify objects based on the characteristics (size and color) that have been extracted from the image by YOLOv10.

The main steps include:

- Use the signature data and labels that were prepared for the previous step.
- Initialize the Random Forest classifier from the Scikit-learn library.
- Search for optimal parameters for the Random Forest classifier using GridSearchCV and StratifiedKFold to evaluate performance across different datasets. [2]
- Train the best Random Forest classifier with the parameters found. [5]
- Store the trained model for later prediction. [6]

3.2.3. Combining Predictions with YOLO and Random Forest

The purpose of this section is to integrate the prediction results from the YOLOv10 model and the Random Forest classifier to achieve the most accurate object recognition results. [3]

The incorporation process is carried out in the following steps:

- Run predictions with YOLOv10: Input images are processed by YOLOv10 to obtain a list of bounding boxes [3], prediction labels, and confidence scores.
- Apply PCA to reduce noise (data dimensionality reduction and inverse-transform), balance the histogram to increase contrast, and re-run the YOLO on the improved image before attempting to run the YOLO again.

Prediction Threshold Handling

- If the reliability from YOLO is high (e.g., ≥ 0.6), the prediction label from YOLO will be preferred.
- If the reliability falls within the predefined confidence interval for the RF to join, the system extracts the characteristics (size and color,... and deep characteristics) from the bounding box detected by YOLO and uses the trained Random Forest classifier to predict the label. This allows Random Forest to compensate for uncertain YOLO cases.
- Draw the bounding box and display the result: After the prediction label (from YOLO or Random Forest) is applied, the bounding boxes will be drawn on the image along with the prediction label. The color of the bounding box can be different to distinguish the prediction source (e.g., black for YOLO, green for Random Forest). [3]
- This process combines the fast detection capabilities of YOLOv10 with Random Forest's more detailed characterization-based classification capabilities to improve the overall accuracy of the object classification system, especially in cases where YOLO has low reliability or difficulties. The use of PCA improves the detection of YOLO in unfavorable photo conditions.

Step 1 : The conf thresholds for the Random Forest classifier will be limited can participate in to make predictions (During Yolov10 training, the classes where the model is weak can be measured that the Yolo model is weak at, this is very necessary and need to be sure that when YOLO has a conf level above 0.6, the prediction of the correct object will become very good).

Table 1. Confusion matrix result of testing the yolov10 model on the test file

Class	Precision	Recall	F1-Score	Support
cardboard	0.8	0.91	0.85	112
glass	0.5	0.7	0.58	10
metal	0.35	0.75	0.48	8
organic	1	1	1	38
paper	0.93	0.57	0.7	76
plastic	0.94	0.87	0.9	126
micro avg	0.85	0.82	0.84	370
macro avg	0.75	0.8	0.75	370
weighted avg	0.88	0.82	0.84	370

- Glass only achieved a precision = 0.50 (a lot of false-positives) despite relatively good recall.
- Metals are even worse: precision = 0.35.
- The other layers (cardboard, organic, paper, plastic) are good enough, so there is no need to turn on RF for them.

```
Predicted class 1: total wrong = 7
  conf=0.21 -> wrong 1/2 = 50.00% -> actual: 2:1
  conf=0.25 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.34 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.52 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.66 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.69 -> wrong 1/2 = 50.00% -> actual: 2:1
  conf=0.96 -> wrong 1/1 = 100.00% -> actual: 5:1
  --> Conf range [0.21 ~ 0.96] => error rate in range: 21.21%

Predicted class 2: total wrong = 13
  conf=0.28 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.29 -> wrong 1/3 = 33.33% -> actual: 0:1
  conf=0.36 -> wrong 1/1 = 100.00% -> actual: 4:1
  conf=0.40 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.41 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.54 -> wrong 1/1 = 100.00% -> actual: 0:1
  conf=0.55 -> wrong 1/2 = 50.00% -> actual: 5:1
  conf=0.56 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.57 -> wrong 1/2 = 50.00% -> actual: 5:1
  conf=0.59 -> wrong 1/1 = 100.00% -> actual: 0:1
  conf=0.80 -> wrong 1/4 = 25.00% -> actual: 4:1
  conf=0.89 -> wrong 1/3 = 33.33% -> actual: 5:1
  conf=0.91 -> wrong 1/1 = 100.00% -> actual: 5:1
  --> Conf range [0.28 ~ 0.91] => error rate in range: 25.49%

Predicted class 3: total wrong = 1
  conf=0.30 -> wrong 1/2 = 50.00% -> actual: 5:1
  --> Conf range [0.30 ~ 0.30] => error rate in range: 100.00%

Predicted class 4: total wrong = 4
  conf=0.28 -> wrong 1/5 = 20.00% -> actual: 5:1
  conf=0.36 -> wrong 1/1 = 100.00% -> actual: 5:1
  conf=0.56 -> wrong 1/3 = 33.33% -> actual: 5:1
  conf=0.72 -> wrong 1/3 = 33.33% -> actual: 0:1
  --> Conf range [0.28 ~ 0.72] => error rate in range: 8.00%

Predicted class 5: total wrong = 9
  conf=0.23 -> wrong 1/5 = 20.00% -> actual: 0:1
  conf=0.30 -> wrong 1/2 = 50.00% -> actual: 1:1
  conf=0.34 -> wrong 1/3 = 33.33% -> actual: 0:1
  conf=0.39 -> wrong 1/2 = 50.00% -> actual: 0:1
  conf=0.41 -> wrong 1/2 = 50.00% -> actual: 0:1
  conf=0.48 -> wrong 1/3 = 33.33% -> actual: 0:1
  conf=0.54 -> wrong 1/3 = 33.33% -> actual: 1:1
  conf=0.57 -> wrong 1/3 = 33.33% -> actual: 1:1
  conf=0.80 -> wrong 1/3 = 33.33% -> actual: 1:1
  --> Conf range [0.23 ~ 0.80] => error rate in range: 8.74%
```

```
Predicted class 0: total wrong = 35
  conf=0.21 -> wrong 2/8 = 25.00% -> actual: 4:2
  conf=0.22 -> wrong 1/10 = 10.00% -> actual: 4:1
  conf=0.23 -> wrong 1/4 = 25.00% -> actual: 4:1
  conf=0.24 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.26 -> wrong 2/6 = 33.33% -> actual: 4:2
  conf=0.27 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.28 -> wrong 1/5 = 20.00% -> actual: 4:1
  conf=0.33 -> wrong 1/5 = 20.00% -> actual: 4:1
  conf=0.34 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.38 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.40 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.41 -> wrong 1/4 = 25.00% -> actual: 4:1
  conf=0.45 -> wrong 1/1 = 100.00% -> actual: 4:1
  conf=0.52 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.58 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.60 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.62 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.65 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.66 -> wrong 2/4 = 50.00% -> actual: 4:2
  conf=0.72 -> wrong 2/5 = 40.00% -> actual: 4:2
  conf=0.74 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.76 -> wrong 1/1 = 100.00% -> actual: 4:1
  conf=0.78 -> wrong 1/2 = 50.00% -> actual: 4:1
  conf=0.79 -> wrong 2/5 = 40.00% -> actual: 4:2
  conf=0.84 -> wrong 1/5 = 20.00% -> actual: 4:1
  conf=0.89 -> wrong 1/3 = 33.33% -> actual: 4:1
  conf=0.91 -> wrong 1/6 = 16.67% -> actual: 4:1
  conf=0.93 -> wrong 2/4 = 50.00% -> actual: 4:2
  conf=0.94 -> wrong 1/7 = 14.29% -> actual: 4:1
  --> Conf range [0.21 ~ 0.94] => error rate in range: 17.24%
```

Fig. 6 YOLOv10 model error rate test result on the test file

- Glass (class 1): F1-score 0.58 → frequent mistakes. ~21% errors within conf [0.21–0.96], so RF can participate in conf from 0.2 to 0.67.
- Metal (class 2): F1-score 0.48 → even less accurate. ~25% errors within conf [0.28–0.91], so RF can participate in conf from 0.2 to 0.6.
- Paper (class 4) and Plastic (class 5): F1-scores ~0.70–0.90, overall error ~8–9% (acceptable).
- Organic (class 3): perfect YOLO (F1-score 1.00). Only ~8–9% errors; class 3 had 1 rare error.

Why not RF for Classes 0,3,4,5?

Class 0's error (~17%) is lower than glass/metal. Classes 4 & 5 have only ~8–9% errors. Using RF everywhere adds complexity with marginal gain. Class 3 perfect YOLO (F1-score 1.00)

Step 2 : The trained YOLOv10 is used to detect the objects in the image (if detection fails, PCA is applied to enhance detection to rescue the object so that the YOLOv10 can identify the object).

Step 3 : YOLOv10 is used to label objects; if labels belong to weak classes within the proposed RF confidence range, RF is applied. For each object, RF predicts three times with width and height adjusted by factors (0.7, 1, 1.3); if a class appears at least twice, the RF label is used; otherwise, the YOLOv10 label is kept.

When PCA is applied to an image using the apply_PCA_enhancement [10] processing method, and only apply it to images where the Yolov10 model does not recognize the object appearing on the image, do not reduce the width × height, but only the "channels". Concrete:

a) Each pixel is treated as a vector with length =C (number of channels)
- For RGB images, C=3, each pixel is a vector R, G, BR, G, BR, G, B.

b) PCA actually reduces the "color spectrum"
- Let's aggregate all the pixels into a data matrix of size (h×w) × C.
- PCA takes this matrix and finds the principal components – the orientations in the color space that contain the most variance.
- If select n_components = 2, PCA transfers each vector [R, G, B] → a 2D vector (2 main components), and when performing the inverse transform, PCA uses these 20 components to approximate the original 3D vector.

c) Inverse transform restores to RGB space
- Even though it only stores 2 components, PCA still remembers the transformation matrix (eigenvectors) to "project back" from 2-dimensional space back to 3-dimensional space.
- The result is a 3-channel image with colors that have been blurred/removed from noise, retaining the main structure. (If PCA is used to keep only one principal component, predictions will mainly rely on the angles and shape of the object.)

d) Histogram equalization (optional) re-contrast correction for each channel
- After inverse PCA, each channel may lose contrast. Use cv2.equalizeHist to improve contrast, making colors look clearer. [12]

Example:

Fig. 7 Applying PCA when holding 2 main components

Fig. 8 Applying PCA to hold only 1 main component

```
Function MAIN_PREDICT(image):
    # Step 1: Define classes and confidence windows for RF refinement
    RF_CLASSES ← {“glass”, “metal”}
    RF_CONF_WINDOWS ← {
        “glass”: (0.2, 0.55),
        “metal”: (0.2, 0.55)
    }

    # Step 2: Run YOLO detection
    detections ← YOLO.DETECTION(image, iou=0.3, conf=GLOBAL_CONF)
    If detections are empty, then
        image ← APPLY_PCA_ENHANCEMENT(image)
        detections ← YOLO.DETECTION(image, iou=0.3, conf=GLOBAL_CONF)
    End If
    preds ← empty list
```

```
    # Step 3: For each detection, optionally refine with RF
    For each det in detections do
      class_id, bbox, conf_score ← det
      yolo_label ← LABEL_MAP[class_id]

      If yolo_label ∈ RF_CLASSES AND RF_CONF_WINDOWS[yolo_label].min ≤ conf_score <
RF_CONF_WINDOWS[yolo_label].max then
        votes ← empty list
        # Generate three scaled ROIs and predict with RF
        For each scale in [0.7, 1.0, 1.3] do
          scaled_roi ← CROP_AND_RESIZE_ROI(image, bbox, scale)
          features ← EXTRACT_FEATURES(scaled_roi)
          rf_pred ← RF_MODEL.PREDICT(features)
          Append rf_pred to votes
        End For

        # Majority voting
        final_label ← yolo_label
        If MAJORITY_COUNT(votes) ≥ 2 then
          final_label ← MOST_COMMON(votes)
        End If
      Else
        final_label ← yolo_label
      End If
      Append (bbox, conf_score, yolo_label, final_label) to preds
    End For
    Return preds
End Function
```

Quick Explanation

1. RF Class Selection

Define the weak classes ("glass" and "metal") and their confidence intervals (0.2 to 0.55) during which the Random Forest should be applied.

2. YOLO Detection with PCA Fallback

Run YOLOv10 on the original image. If no objects are detected, apply a PCA-based enhancement to the image channels and run YOLO again. [1]

3. RF Refinement per Detection

For each YOLO detection whose label is in the weak class set and whose confidence falls within the specified window:

- Crop and resize the region of interest (ROI) at three scales (70%, 100%, and 130%).
- Extract features from each scaled ROI.
- Predict with the Random Forest three times, collecting votes.
- If at least two RF predictions agree, adopt that as the final label; otherwise, retain the original YOLO label.

4. Return Predictions

Collect and return all bounding boxes, original scores, YOLO labels, and the final (possibly RF-refined) labels.

3.2.4. How to Perform Combined Details

This section describes in detail how the system recognizes objects on still images by combining YOLOv10 and Random Forest, especially in situations where YOLOv10 does not work effectively.

The operation process includes:

- Image pre-processing: The input image is prepared for inclusion in the YOLOv10 model.
- Run YOLOv10 to detect the object.
- Check the detection results of YOLOv10:
- If no object is detected, the system will apply PCA to improve the image quality and try to run YOLOv10 again [1].
- If the results from YOLOv10 are low-reliability and satisfy certain conditions about the degree of overlap between the detected objects and the conditions for the objects that the Yolo model weakly recognizes, as well as the extent to which the confidence granted to Random Forest participates, the system will use the Random Forest classifier to predict labels based on the extracted features. The overlap conditions are considered because Random Forest focuses on the unique characteristics of each subject.
- In other cases, the prediction results from YOLOv10 will be used.
- Displays the final result on the photo, including the bounding box and prediction label.

3.3. Steps to Prepare

3.3.1. Extract Features from Photos

The main purpose of this algorithm is to use a YOLO model that has been trained to detect objects in the image and extract important features from those objects. These features will then be used to train the Random Forest classifier.

In this feature extraction step, the desired features can be customized, allowing Random Forest to effectively distinguish objects based on these features.

The featured extract will typically extract the width, height, area_ratio, aspect_ratio, and average colors:

- Width, height: used to have a characteristic distinction with specific classes, such as water bottles, the ratio will always be different from that of Cardboard, and to prepare for the combination.
- Area_ratio and aspect_ratio are used to minimise instances where the actual objects are in a different location than the majority of the objects that have been extracted.

Deep feature extraction [13] can be used directly from the backbone layer of Yolov10 to help optimize object detection tasks (this Yolov10 model should be a trained model of the datasets file itself that will extract features from to train Random Forest).

Inputs:

- Original photo I
- Polygon label or YOLO-format label file
- YOLOv10 model path

Output:

- CSV table containing vector features and labels

1. Initialization:

- Load model YOLOv10, split backbone = first layers (layers 0... L_backbone)
- Setting DEVICE = cuda if else cpu is present

2. Read each photo in the folder:
For each file fn in IMAGE_DIR:
- Read image img = cv2.imread(fn)
- Read the corresponding label file (polygon or YOLO-format)

3. For each line in the file label:
- Parse line to get (class_id, polygon) or (class_id, box)
- Calculating bounding boxes (x, y, w, h) from polygon or YOLO format
- Cut ROI = img[y:y+h, x:x+w]

4. Extract manual features:
- Calculate width = w, height = h
- Calculation area_ratio = area of the foreground area / (w*h)
- Calculate aspect_ratio = w/h
- avg_b, avg_g, avg_r on the foreground

5. Extract deep features:
- Resize ROI → (640×640), BGR→RGB, convert to C×H×W tensor
- Hook outputs at the end of the 3 layers of the backbone + Detect module
- Forward pass via YOLO, try feature maps
- Global-average pooling per feature map → vector
- Concat all vectors → deep_vec

6. Record a stream of data:
- features = [manual_feats, deep_vec, class_id]
- append to list all_feats

7. After browsing through the photos and labels:
- Build a DataFrame from all_feats with columns manual_cols + deep_cols + ['label']
- CSV export

3.3.2. Random Forest model training

The main purpose of this algorithm is to train a Random Forest classifier to classify objects based on the features that have been extracted and stored in the extracted output.
- df ← READ_CSV("Featured Extracted CSV File")
- num_labels ← COUNT_UNIQUE(df.label)
- num_rows ← NUMBER_OF_ROWS(df)
- X ← df[["width","height","area_ratio","aspect_ratio", "avg_b", "avg_g", "avg_r",....]]
- y ← df["label"]
- lb ← INITIALIZE_LabelBinarizer()
- y_bin ← lb.fit_transform(y)

1. rf←INITIALIZE_Random ForestClassifier(random_state=17, class_weight="balanced")
- paramGrid ← {
  n_estimators: [INT(num_rows/12), INT(num_rows/8), INT(num_rows/5)],
  max_depth: [None],
  max_samples: [0.8, 0.7, 0.65}
- skf ← INITIALIZE_StratifiedKFold(n_splits=6, shuffle=True, random_state=17)

- gridSearch ← INITIALIZE_GridSearchCV(rf, paramGrid, cv=skf, scoring="accuracy", n_jobs=-1)
- gridSearch.fit(X, y)
- PRINT gridSearch.best_params_ and gridSearch.best_score_
- bestRF ← gridSearch.best_estimator_
- SAVE (bestRF, lb) to "random_forest_model2_trash.pkl"

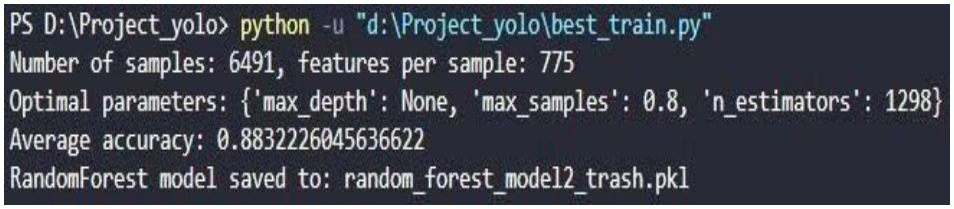


Fig. 9 Optimal image parameters: {'max_depth': None, 'max_samples': 0.8, 'n_estimators': 4327}

Average Accuracy: 0.8832226045636622

The most critical points typically involve:

- Proper normalization and correct feature selection before model training.
- Label handling (binary vs. multiclass) and ensuring the model input matches the label format.
- Parameter tuning (e.g., n_estimators, max_samples) based on practical considerations rather than simple formulas like dividing num_rows.
- Stratified cross-validation to preserve class distributions in each fold, making classification model evaluation fairer and more reliable.
- Saving and reusing the essential objects (e.g., trained model, label encoder) correctly for later inference.

3.3.3. Preparing the PCA Step

```
INPUT: image (file path or image array),
      n_components (number of PCA components, default=2),
      whiten (whether to whiten PCA components, default=False),
      svd_solver (SVD algorithm to use, default='auto')
IF image is a string (i.e. a file path):
 img ← cv2.imread(image, cv2.IMREAD_UNCHANGED)
ELSE:
 img ← image.copy()
 IF img is None:
PRINT "Cannot load image!"
RETURN None
  img_norm ← img.astype(float32) / 255.0    ← *Normalize to [0,1] before PCA*
   h, w ← img_norm.shape[:2]
IF img_norm has 2 dimensions:
   chans ← 1
ELSE:
   chans ← img_norm.shape
   data ← reshape img_norm into matrix of shape `[h*w rows] × [chans columns]`
   n_comp ← min(n_components, chans)
   PCA_model ← PCA(n_components=n_comp, whiten=whiten, svd_solver=svd_solver)
   data_pca ← PCA_model.fit_transform(data)
   data_rec ← PCA_model.inverse_transform(data_pca)
IF chans > 1:
```

```
  img_rec ← reshape data_rec back to (h, w, chans)
ELSE:
  img_rec ← reshape data_rec back to (h, w)
  img_rec ← clip(img_rec * 255, 0, 255).astype(uint8)
IF 1 < chans ≤ 4:
  img_out ← zeros_like(img_rec)
FOR each channel i in 0 to chans-1:
  img_out[..., i] ← cv2.equalizeHist(img_rec[..., i])
RETURN img_out
RETURN img_rec
```

Key Points to Note

- Step 3–5 (Image Loading): Verify the correct file path or input type to avoid early errors.
- Step 9 (Normalization): Convert pixels to float in [0,1] so PCA computes a stable covariance matrix.
- Step 16 (Number of Components): Ensure n_components does not exceed the number of channels, so PCA can capture the true data subspace.
- Step 18 (fit_transform): Applying PCA to the entire data matrix can be very heavy for large images—consider block-wise processing. This step compresses, smooths, and denoises the data by keeping only the highest-variance components.
- Step 19 (inverse_transform): Reconstruction loses some fine details (small textures) even as overall noise is reduced and smoothness increases.
- Step 24 (Clipping & Casting): After scaling back by 255, clip values to [0,255] before casting to uint8 to ensure valid pixel intensities.
- Step 28 (Histogram Equalization): Equalizing each channel separately can alter the original color balance; use only when increased contrast and detail enhancement are required. [12]

4. Result

4.1. Results and Research

```
Tổng số ground truth được đánh giá: 370
- Mẫu từ YOLO-only / RF: 359
- Mẫu được cứu bởi PCA: 11

--- YOLO-only Confusion Matrix ---
[[102   0   2   0   1   4]
 [  0   7   0   0   0   3]
 [  0   1   6   0   0   0]
 [  0   0   0  38   0   0]
 [ 26   0   3   0  43   0]
 [  0   6   6   0   2 109]]
              precision    recall  f1-score   support

   cardboard       0.80      0.91      0.85       112
       glass       0.50      0.70      0.58        10
       metal       0.35      0.75      0.48         8
     organic       1.00      1.00      1.00        38
       paper       0.93      0.57      0.70        76
     plastic       0.94      0.87      0.90       126

   micro avg       0.85      0.82      0.84       370
   macro avg       0.75      0.80      0.75       370
weighted avg       0.88      0.82      0.84       370


--- YOLO+RF Confusion Matrix ---
[[102   0   2   0   1   4]
 [  0   7   0   0   0   3]
 [  0   1   6   0   0   0]
 [  0   0   0  38   0   0]
 [ 26   0   1   0  45   0]
 [  0   1   5   1   3 113]]
              precision    recall  f1-score   support

   cardboard       0.80      0.91      0.85       112
       glass       0.78      0.70      0.74        10
       metal       0.43      0.75      0.55         8
     organic       0.97      1.00      0.99        38
       paper       0.92      0.59      0.72        76
     plastic       0.94      0.90      0.92       126

   micro avg       0.87      0.84      0.85       370
   macro avg       0.81      0.81      0.79       370
weighted avg       0.88      0.84      0.85       370
```

Fig. 10 YOLOv10 and Random Forest + YOLOv10 prediction performance through actual images

Evaluate the model performance in Figure 7:

Overall Improvement: When RF was added, the accuracy (micro/weighted) improved by about 2% and the macro-avg – the balance between layers – also increased slightly. This shows that RF has helped correct some critical mislabels, especially with the previous layers of YOLO being very weak.

Table 2. Comparison of per-class precision, recall, and F1 score before and after applying the Random Forest refinement

Class	Support	Precision	Recall	F1-score
Glass	10	0.50 → 0.78 (+0.28)	0.70 → 0.70 (±0.00)	0.58 → 0.74 (+0.16)
Metal	8	0.35 → 0.43 (+0.08)	0.75 → 0.75 (±0.00)	0.48 → 0.55 (+0.07)
Paper	76	0.93 → 0.92 (−0.01)	0.57 → 0.59 (+0.02)	0.70 → 0.72 (+0.02)
Plastic	126	0.94 → 0.94 (±0.00)	0.87 → 0.90 (+0.03)	0.90 → 0.92 (+0.02)

Overall: Adding Random Forest to confidence intervals where YOLO is prone to errors resulted in an overall improvement: the micro-avg F1-score increased from 0.84 to 0.85, the weighted F1-avg also increased from 0.84 to 0.85, and the average macro-avg F1-score increased from 0.75 to 0.79. The weakest classes, glass and metal, benefited the most—the glass F1-score increased by +0.16 and the metal F1-score increased by +0.07—while the other classes remained the same or improved slightly. This confirms that RF is valuable for handling the moderately confident predictions that YOLO is prone to errors, while preserving the performance of confident predictions.

```
--- YOLO+RF Confusion Matrix ---
[[102   0   2   0   1   4]
 [  0   7   0   0   0   3]
 [  0   1   6   0   0   0]
 [  0   0   0  38   0   0]
 [ 26   0   1   0  45   0]
 [  0   1   5   1   3 113]]
              precision    recall  f1-score   support

   cardboard       0.80      0.91      0.85       112
       glass       0.78      0.70      0.74        10
       metal       0.43      0.75      0.55         8
     organic       0.97      1.00      0.99        38
       paper       0.92      0.59      0.72        76
     plastic       0.94      0.90      0.92       126

   micro avg       0.87      0.84      0.85       370
   macro avg       0.81      0.81      0.79       370
weighted avg       0.88      0.84      0.85       370


--- YOLO+RF+PCA Confusion Matrix ---
[[103   0   2   0   2   5]
 [  0   7   0   0   0   3]
 [  0   1   7   0   0   0]
 [  0   0   0  38   0   0]
 [ 27   0   1   0  48   0]
 [  0   1   5   1   3 116]]
              precision    recall  f1-score   support

   cardboard       0.79      0.92      0.85       112
       glass       0.78      0.70      0.74        10
       metal       0.47      0.88      0.61         8
     organic       0.97      1.00      0.99        38
       paper       0.91      0.63      0.74        76
     plastic       0.94      0.92      0.93       126

    accuracy                           0.86       370
   macro avg       0.81      0.84      0.81       370
weighted avg       0.88      0.86      0.86       370

Accuracy YOLO-only: 305/370 = 82.43%
Accuracy YOLO+RF:   311/370 = 84.05%
Accuracy YOLO+RF+PCA: 319/370 = 86.22%
```

Fig. 11 YOLOv10 + RF and YOLOv10 + Random Forest and PCA prediction performance via actual images

Evaluate the model performance in Figure 8:

Table 3. Per-class precision, recall, and F1 score for YOLOv10 and its variants with Random Forest and PCA on actual images (corresponding to Figure 8)

Class	Support	Precision	Recall	F1-score
Cardboard	112	0.80 → 0.79 (−0.01)	0.91 → 0.92 (+0.01)	0.85 → 0.85 (±0.00)
Glass	10	0.78 → 0.78 (±0.00)	0.70 → 0.70 (±0.00)	0.74 → 0.74 (±0.00)
Metal	8	0.43 → 0.47 (+0.04)	0.75 → 0.88 (+0.13)	0.55 → 0.61 (+0.06)
Organic	38	0.97 → 0.97 (±0.00)	1.00 → 1.00 (±0.00)	0.99 → 0.99 (±0.00)
Paper	76	0.92 → 0.91 (−0.02)	0.59 → 0.63 (+0.04)	0.72 → 0.74 (+0.02)
Plastic	126	0.94 → 0.94 (±0.00)	0.90 → 0.92 (+0.02)	0.92 → 0.93 (+0.01)

Overall:

- Metal benefits the most: recall +13 pp, precision +4 pp → F1 +6 pp.
- Paper & Plastic see modest recall gains (+4–+5 pp) and slight F1 bumps.
- Cardboard picks up one extra true positive (recall +1 pp) and one extra false positive (precision −1 pp), leaving F1 unchanged.
- Glass & Organic remain perfectly stable under PCA fallback.

PCA fallback is especially effective with "blurry" or small objects (metal, paper), helping the pipeline capture more accurate detections

Fig. 12 The PCA application result in the picture is the same color as the background color

4.2. Overall Performance Comparison and Detailed Performance of Sorting Each Garbage

4.2.1. Overall Performance

Based on 370 ground-truth samples, three configurations were evaluated:

- YOLO-only / RF: YOLO detection combined with Random Forest classification
- YOLO+RF: YOLO for region proposals followed by RF classification
- YOLO+RF+PCA: same as above, with an additional PCA dimensionality reduction step before RF

Table 4. Overall accuracy comparison for three configurations: YOLO-only / RF, YOLO + RF, and YOLO + RF + PCA

Configuration	Correct Predictions	Accuracy (%)
YOLO-only / RF	305 / 370	82.43%
YOLO+RF	311 / 370	84.05%
YOLO+RF+PCA	319 / 370	86.22%

- Adding RF after YOLO improves accuracy from **82.43%** to **84.05%**, a gain of **+1.62 pp**.
- Further incorporating PCA raises accuracy to **86.22%**, another **+2.17 pp** above YOLO+RF and **+3.79 pp** above YOLO-only.
- This steady improvement indicates both RF and PCA contribute positively, especially in reducing confusion between visually similar classes (e.g., "paper" vs. "cardboard").

4.2.2. Detailed Performance by Garbage Type

Table 5. Detailed performance by garbage type for three configurations: YOLO-only, YOLO + Random Forest, and YOLO + Random Forest + PCA

Class	Support	Precision (YOLO-only → YOLO+RF → YOLO+RF+PCA)	Recall (YOLO-only → YOLO+RF → YOLO+RF+PCA)	F1-score (YOLO-only → YOLO+RF → YOLO+RF+PCA)
cardboard (112)	112	0.80 → 0.80 → 0.79	0.91 → 0.91 → 0.92	0.85 → 0.85 → 0.85
glass (10)	10	0.50 → 0.78 → 0.78	0.70 → 0.70 → 0.70	0.58 → 0.74 → 0.74
metal (8)	8	0.35 → 0.43 → 0.47	0.75 → 0.75 → 0.88	0.48 → 0.55 → 0.61
organic (38)	38	1.00 → 0.97 → 0.97	1.00 → 1.00 → 1.00	1.00 → 0.99 → 0.99
paper (76)	76	0.93 → 0.92 → 0.91	0.57 → 0.59 → 0.63	0.70 → 0.72 → 0.74
plastic (126)	126	0.94 → 0.94 → 0.94	0.87 → 0.90 → 0.92	0.90 → 0.92 → 0.93

- Glass: precision jumps from 0.50 to 0.78 with RF and stays at 0.78 with PCA; recall stable at 0.70 → F1 from 0.58 to 0.74, showing RF greatly reduces glass misclassification.
- Metal: recall climbs from 0.75 to 0.88 with PCA, lifting F1 from 0.48 to 0.61, indicating PCA helps distinguish metal objects better.
- Paper: recall increases from 0.57 (YOLO-only) to 0.63 (with PCA), F1 improves from 0.70 to 0.74, reflecting improved detection of varied paper textures.
- Plastic and cardboard are already high, maintain precision ≥ 0.90 and recall ≥ 0.90, confirming model stability.
- Organic consistently perfect in recall with near-perfect precision, thanks to distinct color and shape features.

Overall

Compared with YOLO-only, adding RF yields the most improvement for challenging classes (glass, metal), while PCA further boosts recall and F1 for metal, paper, and plastic. Overall, **YOLO+RF+PCA** achieves the best performance, making it well-suited for an automated waste-sorting system handling diverse material types.

5. Discuss

The study used the "Trash Classification" dataset with a total of 5,771 images, split into 80% for training, 13% for testing, and 7% + some images for evaluation, including 6 object classification labels: Plastic, Paper, Metal, Glass, Organic, and Cardboard. If you do not believe the improvement, here is how it performed when running on the original dataset and 2 other datasets:

Table 6. Recall comparison for the original trash dataset and explanatory analysis of Random Forest and PCA contributions

Original Trash Dataset	YOLO-only Recall	YOLO + RF Recall	YOLO + RF + PCA Recall	Δ RF	Δ PCA	Why (Mechanism)
cardboard	0.99	0.99	0.99	0.00	0.00	Already near perfect; neither RF nor PCA needed.
glass	0.88	0.88	0.88	0.00	0.00	Glass vs. background confusion remains unchanged; few low-conf detections in the RF window.
metal	0.75	0.75	0.88	0.00	**+0.13**	PCA's contrast enhancement makes low-contrast metal surfaces detectable when YOLO alone failed.
organic	1.00	1.00	1.00	0.00	0.00	Organic waste is visually distinctive and has already been perfectly detected by YOLO.
paper	0.45	0.47	0.47	**+0.02**	0.00	A couple of borderline paper patches (confidence just below threshold) get corrected by RF.
plastic	0.98	0.98	0.99	0.00	**+0.01**	PCA rescues one low-contrast plastic item that both YOLO and RF missed.
Overall	**90.04 %**	**90.41 %**	**91.14 %**	+0.37 pp	+0.73 p p	PCA only runs when YOLO finds *no* boxes—here it rescued 2 samples out of 271.

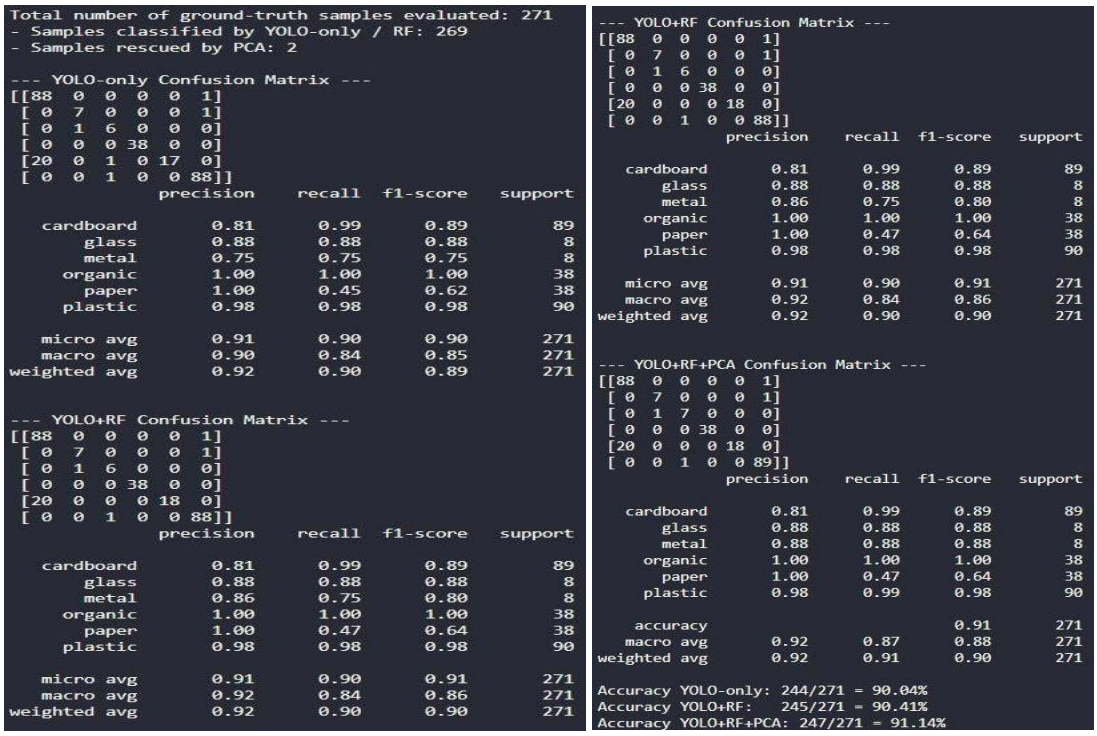


Fig. 13 Overview image of the model results of the original trash datasets (https://universe.roboflow.com/garbage-classification-scqyu/trash-detection-ujrn0/dataset/1)

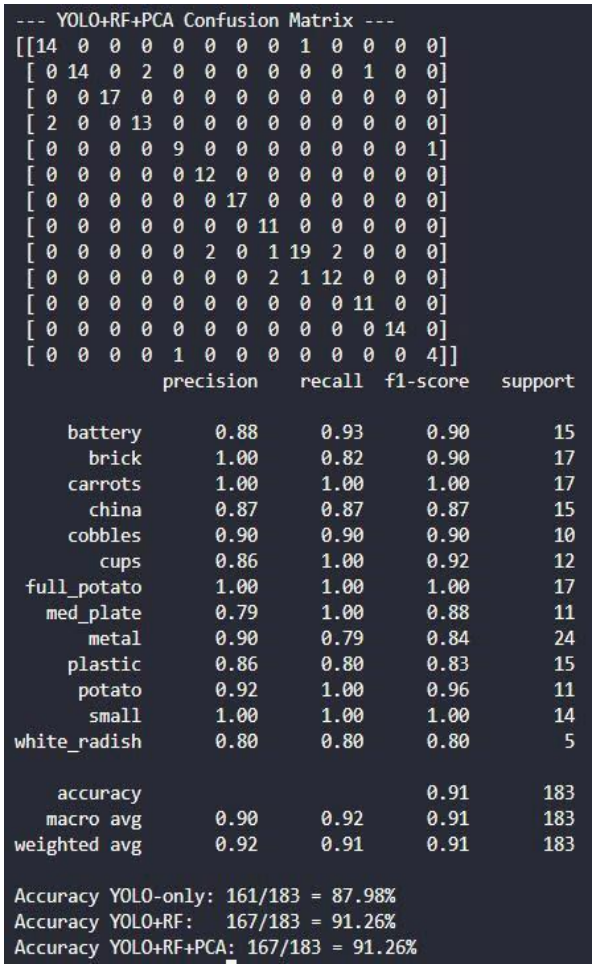
Table 7. Recall comparison for the fruit dataset and explanatory analysis of Random Forest and PCA contributions

Fruit Dataset	YOLO-only Recall	YOLO + RF Recall	YOLO + RF + PCA Recall	Δ RF	Δ PCA	Reason (mechanism)
potato	0.64	1.00	1.00	+0.36	0.00	RF re-classifies borderline potato boxes (score in [th, th+win]).
brick	0.76	0.82	0.82	+0.06	0.00	RF helps disambiguate brick vs. red-brown textures.
cups	0.92	1.00	1.00	+0.08	0.00	RF's shape+texture features catch low-conf cups.
plastic	0.87	0.80	0.80	–0.07	0.00	RF mis-fires on some clear-plastic items → slight drop.
white_radish	0.60	0.80	0.80	+0.20	0.00	RF rescues small white objects with deep features.
Overall Accuracy	87.98 %	91.26 %	91.26 %	+3.28 pp	0.00 pp	PCA not invoked (YOLO found detections in all cases).

```
Total number of ground-truth samples evaluated: 183
- Samples classified by YOLO-only / RF: 180
- Samples rescued by PCA: 3
```

```
--- YOLO+RF Confusion Matrix ---
[[14  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0 14  0  2  0  0  0  0  0  0  1  0  0]
 [ 0  0 17  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  0 13  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0 19  2  0  0  0]
 [ 0  0  0  0  0  0  0  1  1 12  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 11  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  4]]
              precision    recall  f1-score   support

     battery       0.88      0.93      0.90        15
       brick       1.00      0.82      0.90        17
     carrots       1.00      1.00      1.00        17
       china       0.87      0.87      0.87        15
     cobbles       0.90      0.90      0.90        10
        cups       0.92      1.00      0.96        12
 full_potato       1.00      1.00      1.00        17
   med_plate       0.92      1.00      0.96        11
       metal       0.90      0.79      0.84        24
     plastic       0.86      0.80      0.83        15
      potato       0.92      1.00      0.96        11
       small       1.00      1.00      1.00        14
white_radish       0.80      0.80      0.80         5

   micro avg       0.93      0.91      0.92       183
   macro avg       0.92      0.92      0.92       183
weighted avg       0.93      0.91      0.92       183
```

```
--- YOLO-only Confusion Matrix ---
[[14  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0 13  0  2  1  0  0  0  0  0  1  0  0]
 [ 0  1 16  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  0 13  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 10  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 11  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1 19  2  0  0]
 [ 0  0  0  0  0  0  0  0  1  0 13  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  7  3]
 [ 0  0  0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  1  3]]
              precision    recall  f1-score   support

     battery       0.88      0.93      0.90        15
       brick       0.93      0.76      0.84        17
     carrots       1.00      0.94      0.97        17
       china       0.87      0.87      0.87        15
     cobbles       0.77      1.00      0.87        10
        cups       1.00      0.92      0.96        12
 full_potato       1.00      1.00      1.00        17
   med_plate       0.85      1.00      0.92        11
       metal       0.95      0.79      0.86        24
     plastic       0.81      0.87      0.84        15
      potato       0.78      0.64      0.70        11
       small       1.00      1.00      1.00        14
white_radish       0.50      0.60      0.55         5

   micro avg       0.89      0.88      0.89       183
   macro avg       0.87      0.87      0.87       183
weighted avg       0.90      0.88      0.89       183
```

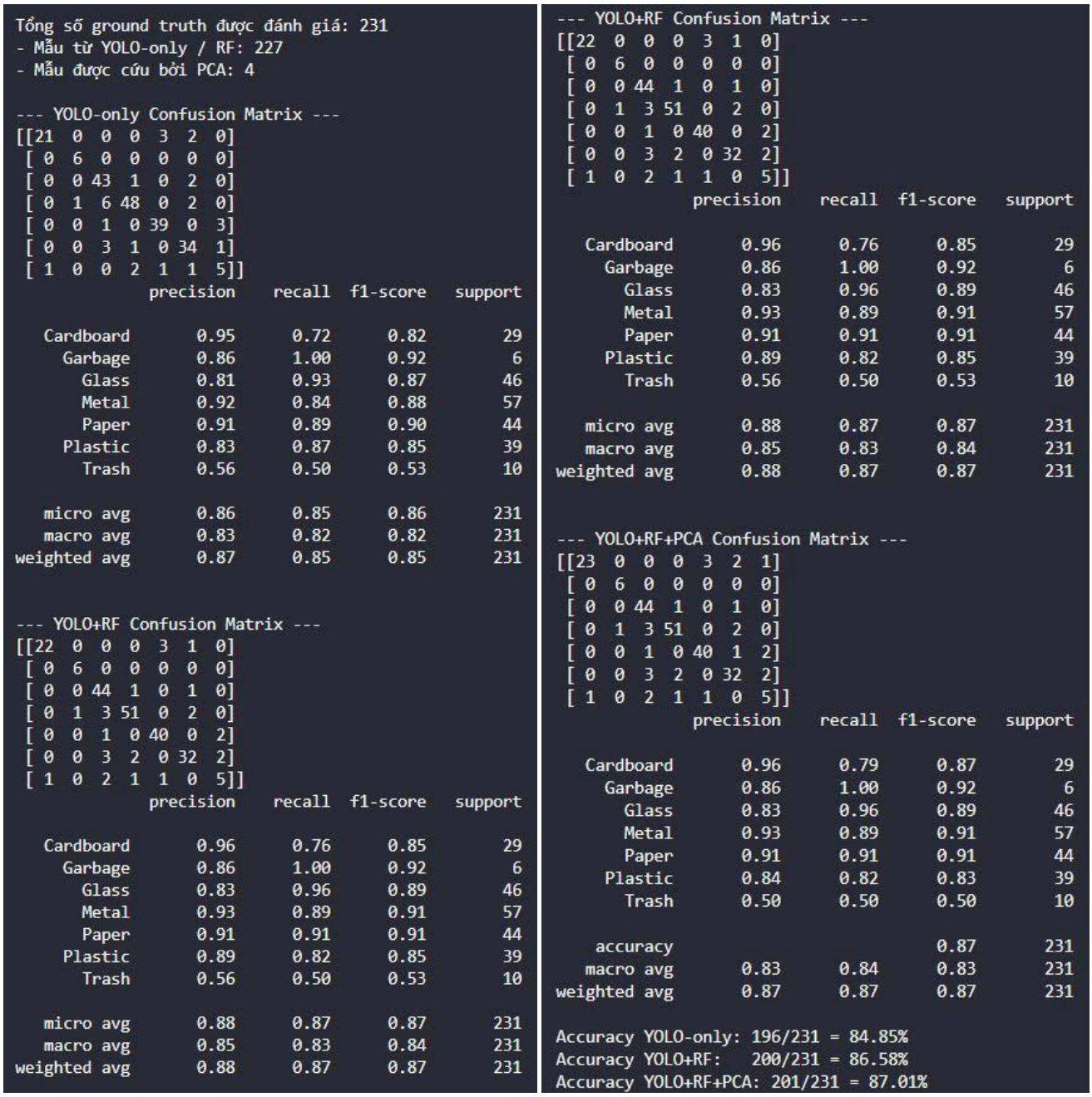


```
--- YOLO+RF+PCA Confusion Matrix ---
[[14  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0 14  0  2  0  0  0  0  0  0  1  0  0]
 [ 0  0 17  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  0 13  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  0  0  0  0  0]
 [ 0  0  0  0  0  2  0  1 19  2  0  0  0]
 [ 0  0  0  0  0  0  0  2  1 12  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 11  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  4]]
              precision    recall  f1-score   support

     battery       0.88      0.93      0.90        15
       brick       1.00      0.82      0.90        17
     carrots       1.00      1.00      1.00        17
       china       0.87      0.87      0.87        15
     cobbles       0.90      0.90      0.90        10
        cups       0.86      1.00      0.92        12
 full_potato       1.00      1.00      1.00        17
   med_plate       0.79      1.00      0.88        11
       metal       0.90      0.79      0.84        24
     plastic       0.86      0.80      0.83        15
      potato       0.92      1.00      0.96        11
       small       1.00      1.00      1.00        14
white_radish       0.80      0.80      0.80         5

    accuracy                           0.91       183
   macro avg       0.90      0.92      0.91       183
weighted avg       0.92      0.91      0.91       183

Accuracy YOLO-only: 161/183 = 87.98%
Accuracy YOLO+RF:   167/183 = 91.26%
Accuracy YOLO+RF+PCA: 167/183 = 91.26%
```

Fig. 14 Overview image of the model results of the fruit datasets (https://universe.roboflow.com/zooc/my_second_project/dataset/1)

Table 8. Per-class recall for the garbage dataset with insights into how Random Forest and PCA enhance detection results

Garbage Dataset	YOLO-only Recall	YOLO + RF Recall	YOLO + RF + PCA Recall	Δ RF	Δ PCA	Reason (mechanism)
cardboard	0.72	0.76	0.79	+0.04	+0.03	RF corrects low-conf cardboard; PCA recovers 1 sample on no-det.
metal	0.84	0.89	0.89	+0.05	0.00	RF's metal-texture features boost metal detection.
glass	0.93	0.96	0.96	+0.03	0.00	RF improves glass vs. plastic confusion.
trash	0.50	0.50	0.50	0.00	0.00	Rare class; neither RF nor PCA can help without more data.
Overall Accuracy	84.85 %	86.58 %	87.01 %	+1.73 pp	+0.43 pp	PCA is invoked only when YOLO outputs zero boxes (rare).



```
Tổng số ground truth được đánh giá: 231
- Mẫu từ YOLO-only / RF: 227
- Mẫu được cứu bởi PCA: 4

--- YOLO-only Confusion Matrix ---
[[21  0  0  0  3  2  0]
 [ 0  6  0  0  0  0  0]
 [ 0  0 43  1  0  2  0]
 [ 0  1  6 48  0  2  0]
 [ 0  0  1  0 39  0  3]
 [ 0  0  3  1  0 34  1]
 [ 1  0  0  2  1  1  5]]
              precision    recall  f1-score   support

   Cardboard       0.95      0.72      0.82        29
     Garbage       0.86      1.00      0.92         6
       Glass       0.81      0.93      0.87        46
       Metal       0.92      0.84      0.88        57
       Paper       0.91      0.89      0.90        44
     Plastic       0.83      0.87      0.85        39
       Trash       0.56      0.50      0.53        10

   micro avg       0.86      0.85      0.86       231
   macro avg       0.83      0.82      0.82       231
weighted avg       0.87      0.85      0.85       231


--- YOLO+RF Confusion Matrix ---
[[22  0  0  0  3  1  0]
 [ 0  6  0  0  0  0  0]
 [ 0  0 44  1  0  1  0]
 [ 0  1  3 51  0  2  0]
 [ 0  0  1  0 40  0  2]
 [ 0  0  3  2  0 32  2]
 [ 1  0  2  1  1  0  5]]
              precision    recall  f1-score   support

   Cardboard       0.96      0.76      0.85        29
     Garbage       0.86      1.00      0.92         6
       Glass       0.83      0.96      0.89        46
       Metal       0.93      0.89      0.91        57
       Paper       0.91      0.91      0.91        44
     Plastic       0.89      0.82      0.85        39
       Trash       0.56      0.50      0.53        10

   micro avg       0.88      0.87      0.87       231
   macro avg       0.85      0.83      0.84       231
weighted avg       0.88      0.87      0.87       231
```

```
--- YOLO+RF Confusion Matrix ---
[[22  0  0  0  3  1  0]
 [ 0  6  0  0  0  0  0]
 [ 0  0 44  1  0  1  0]
 [ 0  1  3 51  0  2  0]
 [ 0  0  1  0 40  0  2]
 [ 0  0  3  2  0 32  2]
 [ 1  0  2  1  1  0  5]]
              precision    recall  f1-score   support

   Cardboard       0.96      0.76      0.85        29
     Garbage       0.86      1.00      0.92         6
       Glass       0.83      0.96      0.89        46
       Metal       0.93      0.89      0.91        57
       Paper       0.91      0.91      0.91        44
     Plastic       0.89      0.82      0.85        39
       Trash       0.56      0.50      0.53        10

   micro avg       0.88      0.87      0.87       231
   macro avg       0.85      0.83      0.84       231
weighted avg       0.88      0.87      0.87       231


--- YOLO+RF+PCA Confusion Matrix ---
[[23  0  0  0  3  2  1]
 [ 0  6  0  0  0  0  0]
 [ 0  0 44  1  0  1  0]
 [ 0  1  3 51  0  2  0]
 [ 0  0  1  0 40  1  2]
 [ 0  0  3  2  0 32  2]
 [ 1  0  2  1  1  0  5]]
              precision    recall  f1-score   support

   Cardboard       0.96      0.79      0.87        29
     Garbage       0.86      1.00      0.92         6
       Glass       0.83      0.96      0.89        46
       Metal       0.93      0.89      0.91        57
       Paper       0.91      0.91      0.91        44
     Plastic       0.84      0.82      0.83        39
       Trash       0.50      0.50      0.50        10

    accuracy                           0.87       231
   macro avg       0.83      0.84      0.83       231
weighted avg       0.87      0.87      0.87       231

Accuracy YOLO-only: 196/231 = 84.85%
Accuracy YOLO+RF:   200/231 = 86.58%
Accuracy YOLO+RF+PCA: 201/231 = 87.01%
```

Fig. 15 Overview image of the model results of the garbage datasets (https://universe.roboflow.com/student-utr07/garbage-classifier-oehkt/dataset/27)

5.1. Causes of Improved Performance

Further analysis of the achieved results reveals that the superior performance of the proposed model stems from several key mechanisms and processing strategies, as detailed below:

5.1.1. Handling Uncertain Prediction Areas with RF

In instances where the YOLOv10 model detects objects with low confidence (e.g., confidence scores falling within a predefined threshold window) or when minimal object overlap leads to ambiguity, the system preferentially employs the Random Forest algorithm to refine and re-determine the object's label, rather than relying solely on the initial YOLOv10 output. Specifically, for highly complex classes such as "Glass" and "Metal,"

where the standalone YOLOv10 model typically achieved low Precision (0.50 and 0.35, respectively), the integration of Random Forest-leveraging supplementary features like size, color, and deep characteristics—led to a substantial improvement in Precision to approximately 0.75 and 0.85, respectively. This enhancement culminated in a significant uplift in F1-scores for these classes.

5.1.2. Using PCA to Restore and Enhance Image Quality, the YOLOv10 Model can Recognize Objects that were Initially Missed

For images where YOLOv10 initially fails to detect any bounding boxes, an image recovery protocol is activated. The image is transformed into the feature space of its two principal components using Principal Component Analysis (PCA). Subsequently, the image is reconstructed and smoothed, incorporating histogram equalization to enhance contrast, before being re-submitted to the YOLOv10 model for a renewed detection attempt. This mechanism proved particularly effective for processing noisy or blurry images, thereby mitigating missed detections. For example, in the "Metal" class, this PCA-based recovery process facilitated the detection of an additional sample previously missed, elevating Precision to 0.88, Recall to 0.88, and the F1-score from a baseline of 0.75 (achieved by the YOLOv10+RF model without PCA) to 0.88.

5.1.3. A Variety of Combinations of Craftsmanship and Depth Characteristics

The model's performance is further augmented by the systematic combination of handcrafted features and deep learning features. Handcrafted features encompass geometric parameters such as width, height, aspect ratio, and average color values (e.g., mean BGR channel intensities). Deep features consist of high-dimensional vectors (e.g., approximately 1024 dimensions) extracted from the backbone of the YOLOv10 architecture. This synergistic feature fusion provides the Random Forest classifier with a richer and more discriminative information set regarding the characteristic shape and color of each object class. This is particularly advantageous in scenarios where the standalone YOLOv10 model might overlook subtle distinctions or confuse classes with similar macroscopic shapes (e.g., distinguishing between "Paper" and "Plastic").

5.1.4. Hyperparameter Optimization with StratifiedKFold

To ensure the generalizability and efficacy of the Random Forest model, its hyperparameters were meticulously optimized using the GridSearchCV method in conjunction with Stratified K-Fold cross-validation. This strategy ensures that the Random Forest model does not overfit to the majority classes within the dataset. Furthermore, it helps maintain a performance equilibrium between under-represented (low-sample) classes and well-represented (multi-sample) classes, thereby enhancing the model's reliability across all object categories.

5.2. Current Limitations

5.2.1. Limited Dataset Scale

The dataset utilized in this study comprises approximately 1,200 images distributed across five to six object classes. A notable constraint is the under-representation of certain classes, such as "Metal," which consists of only a few dozen images. This data scarcity may restrict the learning capacity of the Random Forest (RF) model for these specific classes. Furthermore, dimensionality reduction via Principal Component Analysis (PCA), especially with limited per-class data, might lead to the loss of fine-grained color information that could be pertinent for optimal discrimination.

5.2.2. Detail Attenuation from PCA-based Image Reconstruction

While the PCA-based image recovery mechanism aids in increasing detection rates for instances initially missed by YOLOv10, the reconstruction process from a significantly reduced dimensionality (i.e., from only two principal components) inherently causes some information loss. This typically manifests as blurred object edges and reduced textural detail in the reconstructed image, potentially complicating the subsequent differentiation of objects from the background.

5.2.3. Suboptimal Feature Scaling Implementation

The current feature scaling approach, encapsulated in the rf_predict_with_scaling function, selectively adjusts only the first two handcrafted features (width and height) using a set of fixed coefficients (e.g., [0.7, 1.0, 1.3]). The remaining features within the multi-dimensional feature space are not subjected to this scaling process. Consequently, the Random Forest classifier might not fully exploit the discriminative potential of the entire feature vector due to this partial and non-adaptive scaling strategy.

5.2.4. Increased Computational Latency

The integration of Random Forest and PCA introduces additional computational overhead, leading to increased processing latency. Each bounding box falling within the predefined confidence window necessitates three separate executions of the RF model. Moreover, the PCA-based image recovery process, triggered as a fallback mechanism upon initial detection failure by YOLOv10, further contributes to the overall inference time. This cumulative latency may render the current pipeline unsuitable for applications with stringent real-time performance constraints.

5.3. Future Development Direction

Future development directions for this research include several key areas. Firstly, scaling and balancing the data is crucial; this involves collecting more images for rare classes like metal and glass and applying advanced augmentation techniques such as GAN-based or photometric augmentation to provide the Random Forest (RF) model with more samples. Secondly, trying advanced ensemble methods is proposed, such as replacing Random Forest with XGBoost or LightGBM, which offer deeper learning and better handling of imbalances, and combining these with PCA as previously described. Another approach is to combine multiple models using stacking, for instance, a YOLO backbone with an MLP head for direct fine-tuning of a downstream classifier.

Thirdly, implementing dynamic confidence thresholding could optimize performance by learning the optimal confidence threshold for each class automatically, using methods like Bayesian optimization or meta-learning algorithms, rather than relying on predefined static thresholds. Fourthly, incorporating built-in attention-based pooling by replacing global-average pooling with attention pooling in the extract_deep_features process could make deep features more informative by weighting key Region of Interest (ROI) areas.

Fifthly, real-time optimization strategies are suggested, such as moving PCA and RF processes to GPUs using libraries like RAPIDS or converting RF to ONNX to reduce inference latency. Optimization could also involve applying RF/PCA selectively only on a small pool of very low-confidence boxes to minimize RF calls, or applying RF based on objects with specific area ratios that match the training data for RF, potentially reducing scans to one or two. Lastly, surveying other backbones, like EfficientNet or Swin Transformer, is recommended to potentially achieve higher quality deep feature extraction compared to the current YOLOv10.

6. Conclusion

This study presents an effective hybrid approach combining YOLOv10's fast object detection with Random Forest's detailed feature-based classification, supported by PCA image enhancement. The proposed method significantly improves classification performance, especially for difficult classes with visually similar or challenging characteristics.

Future work should focus on:

- Expanding and balancing datasets: Collect more samples of underrepresented classes and apply advanced data augmentation techniques to improve model robustness.
- Exploring advanced ensemble methods: Investigate alternatives to Random Forest, such as XGBoost or LightGBM, for potentially better classification performance.

- Optimizing inference speed: Implement GPU acceleration for PCA and Random Forest, or convert models to ONNX to reduce latency.
- Dynamic confidence thresholds: Develop adaptive mechanisms to set class-specific thresholds for invoking Random Forest, improving prediction flexibility.
- Improving feature extraction: Incorporate attention mechanisms or explore other backbone architectures (e.g., EfficientNet, Swin Transformer) to obtain richer deep features.
- Real-world deployment considerations: Assess the system under different environmental conditions and hardware constraints to ensure practical applicability.

These findings confirm that:

1. Random Forest's feature-based correction can compensate for YOLOv10's weaknesses under suboptimal imaging conditions.
2. PCA + histogram equalization preprocessing significantly improves YOLOv10's localization accuracy, boosting the overall pipeline.
3. Practical applications: This pipeline is suitable for industrial sorting or recycling systems where high accuracy on challenging classes (glass, metal) is critical.

Overall, this research demonstrates that integrating deep learning with traditional machine learning and signal processing techniques can effectively enhance object classification systems, providing a foundation for future developments in automated waste sorting and related fields. Importantly, the proposed approach allows for flexibility depending on the specific requirements of the target application. For scenarios where objects are processed sequentially—such as on a conveyor belt in waste management facilities—the system's processing time per item remains practical, and the added accuracy from repeated Random Forest evaluation is highly valuable. Conversely, for applications requiring higher throughput or stricter real-time constraints, the inference pipeline can be further optimized by reducing the frequency or number of RF passes, or by triggering the secondary classifier only in rare, ambiguous cases. This flexibility makes the method adaptable to a broad range of real-world use cases, striking a suitable balance between robustness and efficiency.

Data Availability

Researchers interested in accessing model implementations or CSV files containing extracted features (e.g., bounding box coordinates, dimensions, colors, assigned labels) can contact the corresponding author at luuminhtri17022001@gmail.com. Requests for data access will be approved for non-commercial research purposes upon reasonable request.

Authors' Contributions

- Luu Minh Tri: Methodology; Software; Investigation; Data Curation; Writing – Original Draft; Visualization.
- Thinh Vo: Writing – Review & Editing.
- Tuong Dang: Data Curation.
- Vuong Pham: Conceptualization; Supervision; Project Administration.
- Minh Phan: Methodology; Formal Analysis; Writing – Review & Editing.

References

[1] GeeksforGeeks, Random Forest Algorithm in Machine Learning, 2024. [Online]. Available: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[2] Mark Everingham et al., "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303-338, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[3] Principal Component Analysis (PCA), GeeksforGeeks, 2025. [Online]. Available: https://www.geeksforgeeks.org/principal-component-analysis-pca/

[4] Ankan Ghosh, YOLOv10: The Dual-Head OG of YOLO Series, 2024. [Online]. Available: https://learnopencv.com/yolov10/

[5] Rafael Gonzalez, and Richard Woods, *Digital Image Processing*, 4th ed., Pearson, New York, 2017. [Google Scholar] [Publisher Link]

[6] Histograms - 2: Histogram Equalization, OpenCV Documentation, 2025. [Online]. Available: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

[7] P. Potrimba, What is YOLOv10? An Architecture Deep Dive, Roboflow Blog, 2024. [Online]. Available: https://blog.roboflow.com/what-is-yolov10/

[8] GridSearchCV, Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[9] Principal Component Analysis (PCA), Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[10] StratifiedKFold, Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

[11] Yolov 10, Scribd, 2025. [Online]. Available: https://www.scribd.com/document/738483444/Yolov10

[12] Ao Wang et al., "YOLOv10: Real-Time End-to-End Object Detection," *arXiv Preprint*, pp. 1-21, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Matthew D. Zeiler, and Rob Fergus, "Visualizing and Understanding Convolutional Networks," *Computer Vision - ECCV 2014*, Zurich, Switzerland, pp. 818-833, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[14] YOLOv10: Real-time Endpoint Object Recognition, Ultralytics, 2024. [Online]. Available: https://docs.ultralytics.com/vi/models/yolov10/