Garbage Classification System Documentation

بسملة احمد محمد السيد 202203888

رانيا محمد عبدالسميع علي 202203992

فاطمة محمد عبدالوهاب 202203801

منة الله احمد عزت 202202761

مريم محمد عبد الحليم 202201918

محمد ابراهيم ابرهيم عوض202203103

1. Introduction and Problem Statement

Automated waste classification plays a critical role in modern waste management systems, enabling efficient sorting, recycling, and disposal. While deep learning approaches have shown promise in this domain, they often require substantial computational resources and lack interpretability. This implementation addresses these challenges by using traditional machine learning techniques that deliver strong performance while maintaining transparency and efficiency.

The approach consists of three primary components:

1. A specialized image preprocessing pipeline
2. A robust handcrafted feature extraction system
3. A Random Forest classifier for accurate categorization

2. Code and Dataset Information

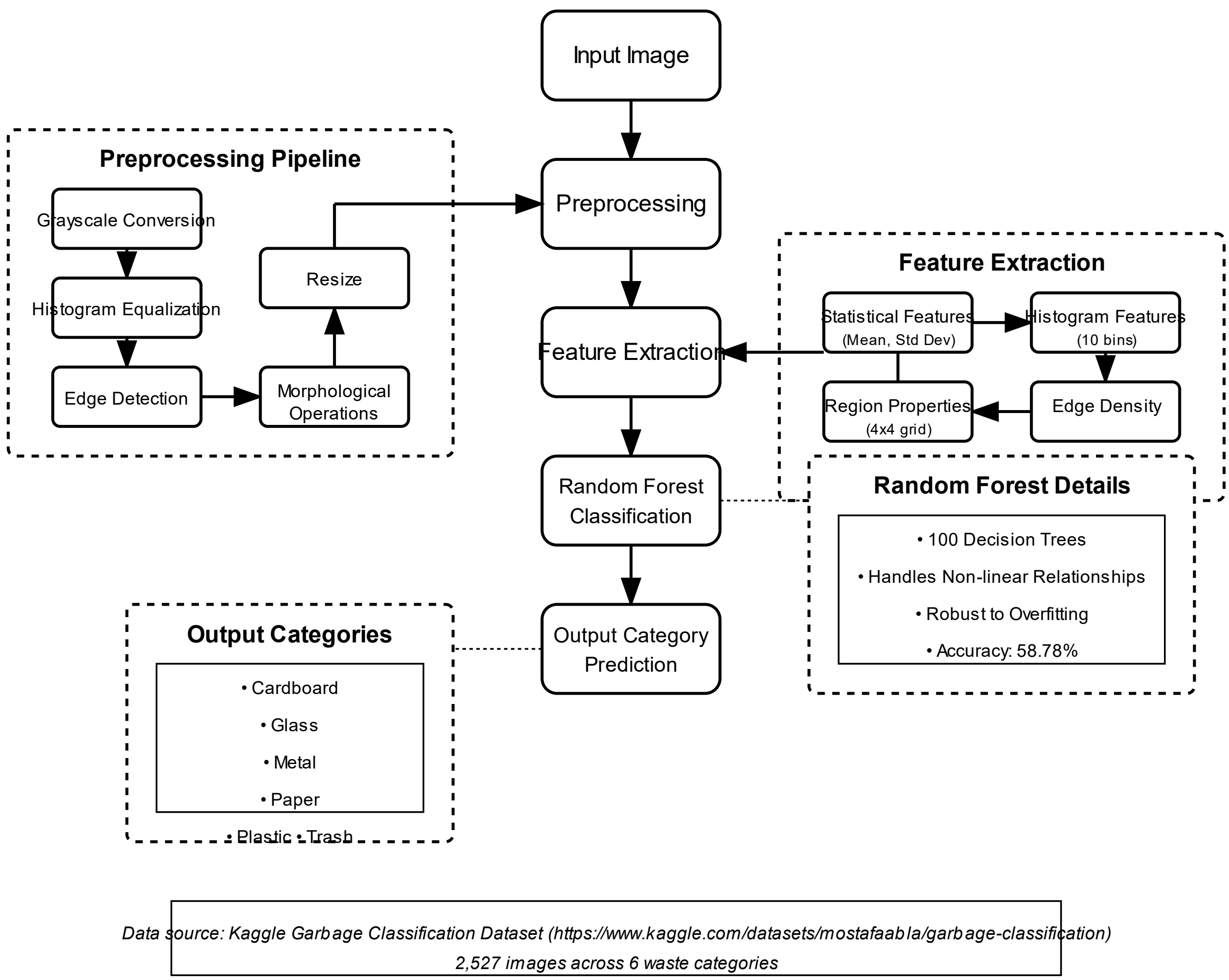
Project link on GitHub

The project utilizes the Garbage Classification dataset from Kaggle, which contains 2,527 images across 6 categories of waste:

- Cardboard
- Glass
- Metal

- Paper
- Plastic
- Biological

3. System Pipeline Overview:



4. Preprocessing Techniques:

The system employs a multi-step image preprocessing pipeline to highlight distinctive features in waste images:

1. ***Grayscale Conversion***
    - Reduces image complexity and computational load
    - Eliminates color variance which can be misleading for material identification
2. ***Histogram Equalization***
    - Enhances contrast in images
    - Makes texture patterns more distinguishable
    - Critical for detecting material surface characteristics
3. **Edge Detection (Sobel)**
    - Highlights boundaries and shape information
    - Captures structural features of different waste materials
4. **Morphological Operations**
    - Strengthens detected edges
    - Removes noise and connects broken edge segments

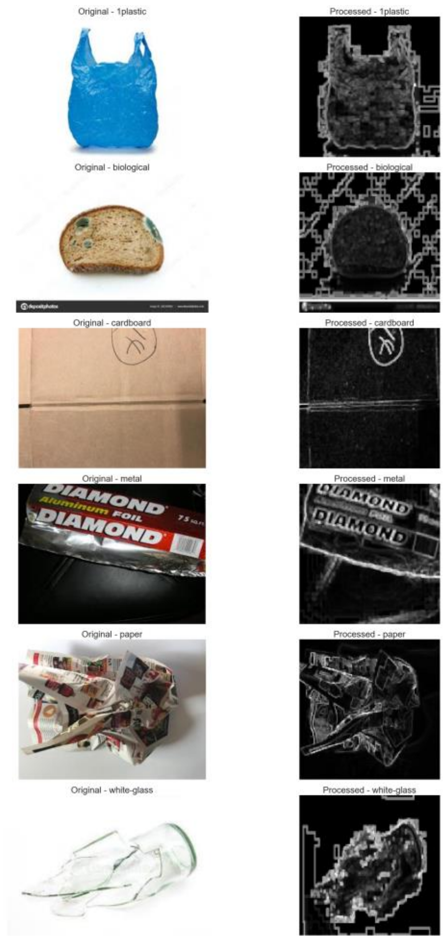
```
# preprocessing functio
def preprocess_imag (img, img_size=(128, 128)):
    e
    # Convert to grayscal
    gray = cv2.cvtColor img, cv2.COLOR_BGR2GRA
              (                  Y)
    # Apply histogram equalizatio
    equalized = cv2.equalizeHist gray)
                   (
    # Edge detection using Sobel operato
    sobel_x = cv2.Sobel(equalized, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(equalized, cv2.CV_64F, 0, 1, ksize=3)
    sobel = cv2.magnitude(sobel_x, sobel_y)
    sobel = cv2.normalize(sobel, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)

    # Morphological operations (dilatio
    kernel = np.ones((3,3), np.uint8)
    dilated = cv2.dilate(sobel, kernel, iteration =1)
                                                 s
    # Resize for model inpu
    resized = cv2.resize(dilated, img_size)

    return resized
```

5. Feature Extraction:

the system extracts handcrafted features for traditional ML classification:

```python
#Feature extraction functio
def extract_feature (img):
    s
    # Simple feature extraction using image statistic
    features = []

    # Mean and standard deviatio
    features.append(np.mean(img))
    features.append(np.std(img))

    # Histogram features (10 bin
    hist = np.histogram(img, bins=10, range=(0, 256))[0]
    features.extend(hist / np.sum(hist))  # Normalize histogra
                                            m
    # Edge density (ratio of edge pixels to total pixel
    edges = cv2.Canny(img.astype(np.uint8), 100, 200)
    features.append(np.sum(edges > 0) / (img.shape[0] * img.shape[1
]))


# Region properties (divide image into 4x4 grid and compute mean of
each region)
    h, w = img.shape
    for i in range(4):
        for j in range(4):
            region = img[i*h//4:(i+1)*h//4, j*w//4:(j+1)*w//4]
            features.append(np.mean(region))

    return features
```

1. **Statistical Features**
   - Mean and standard deviation capture overall brightness and contrast
2. **Histogram Features**
   - 10-bin normalized histogram represents intensity distribution
   - Helps distinguish materials with different brightness patterns
3. **Edge Density**
   - Ratio of edge pixels to total pixels
   - Materials like metal and glass have different edge characteristics than paper or cardboard
4. **Region Properties**
   - Divides image into 4×4 grid and computes mean of each region
   - Captures spatial distribution of features



6. Classification Algorithm

Random Forest Classifier

- Accuracy achieved: 58.78%
- Used with 100 decision trees
- Handles the heterogeneous feature space well
- Robust to overfitting

```
# Cell: Build the Random Forest Classifier mode
print("Training Random Forest Classifie     )
clf = RandomForestClassifie (n_estimator =100, random_stat =42, n_jobs=-1)
clf.fit(X_train, y_train)    s                 e
```

7. Performance Evaluation:

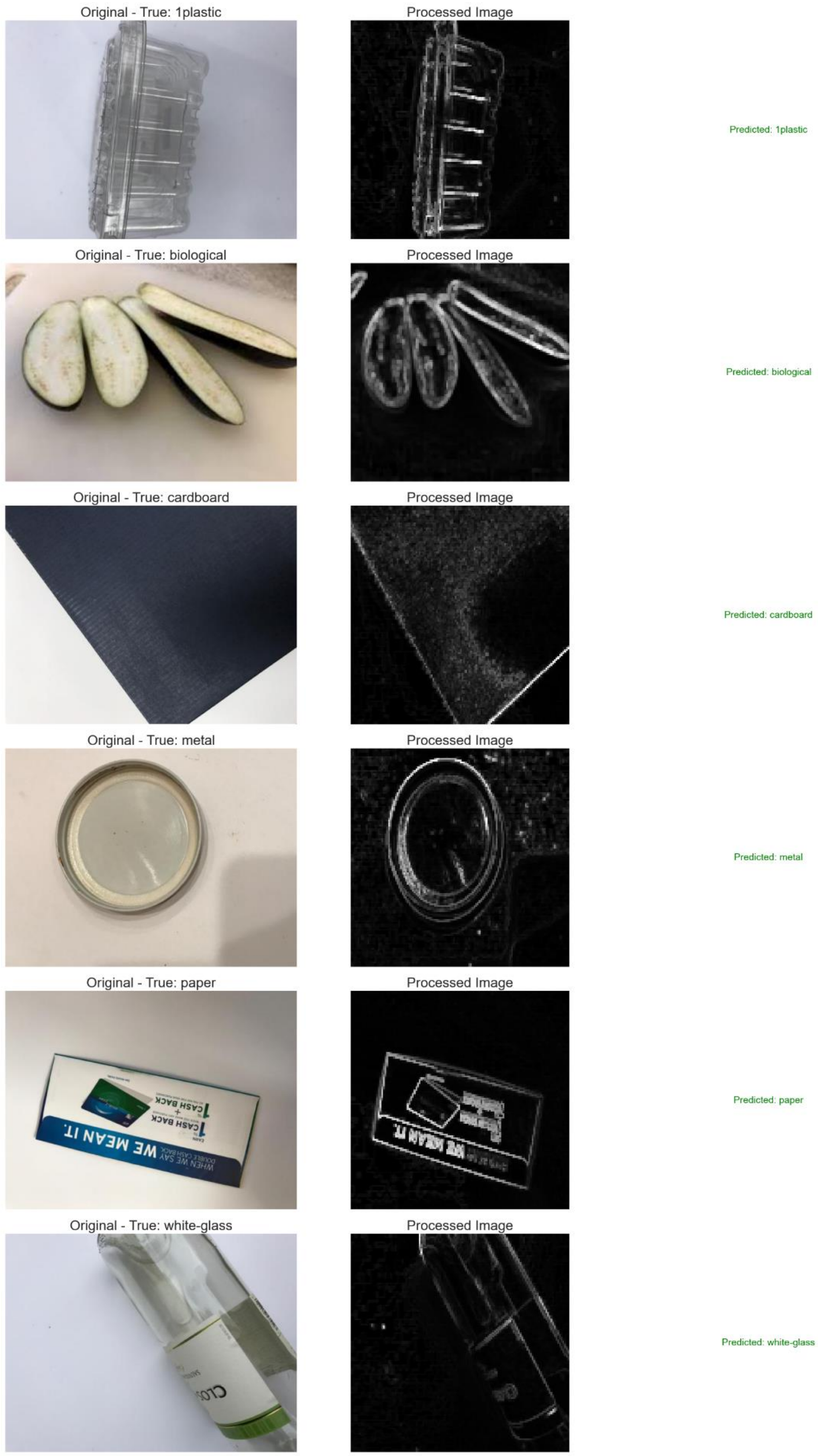
The system evaluates performance using:

- Accuracy score
- Classification report (precision, recall, F1-score)
- Confusion matrix

Classification Report:

	precision	recall	f1-score	support
plastic	0.54	0.41	0.47	198
biological	0.68	0.74	0.71	190
cardboard	0.57	0.69	0.62	173
metal	0.46	0.45	0.45	132
paper	0.65	0.71	0.67	211
white-glass	0.56	0.47	0.51	149
accuracy			0.59	1053
macro avg	0.57	0.58	0.57	1053
weighted avg	0.58	0.59	0.58	1053

Classification Output Sample 1



8. Algorithm Comparison:

Algorithm	Strengths	Limitations	Potential Accuracy
Random Forest (Current)	Handles mixed feature types, resistant to overfitting, captures non-linear relationships	Can be memory-intensive, slower predictions than simpler models	58.78%
SVM	Effective in high-dimensional spaces	Sensitive to feature scaling.	~51-60% with proper tuning
CNN (Deep Learning)	Automatic feature extraction, state-of-the-art performance for image tasks	Requires more data, computationally expensive	70-90% potential

9. Limitations and Future Improvements:

- ### Current Limitations
  - Handcrafted features may miss complex visual patterns
  - Moderate accuracy (58.78%) leaves room for improvement
  - Preprocessing steps might remove some valuable information
- ### Potential Improvements
  - Deep learning approaches like CNNs would likely improve accuracy significantly
  - Additional features like texture analysis (GLCM, LBP)
  - Data augmentation to increase training set size

Our model implementation utilizes Random Forest as the primary algorithm after extensive experimentation. While we evaluated Support Vector Machines (SVM) during the development phase (51.26%), Random Forest consistently delivered superior accuracy metrics across our testing dataset. Moving forward, we plan to enhance classification performance by exploring deep learning approaches through TensorFlow, which may capture more complex feature relationships in waste imagery and potentially improve accuracy beyond what traditional machine learning methods have achieved.