Garbage Classification using Histogram of Oriented Gradients (HOG)

Tanya Budhrani

Hong Kong Polytechnic University

I. Introduction

In our daily lives, proper garbage classification plays a crucial role in environmental preservation and sustainable waste management. Effective waste sorting helps in recycling, reducing pollution, and conserving natural resources. However, manual sorting is inefficient and prone to errors, making automated garbage classification a valuable tool for improving waste management efficiency.

Recent advancements in machine learning and computer vision have therefore sought to automate garbage classification, potentially accelerating waste-handling processes and reducing the likelihood of mixing non-recyclable materials with reusable ones (IBM, 2023). These automated solutions can classify various types of refuse—plastic, paper, cardboard, and even materials like metal or glass—thus promoting more efficient sorting and recycling workflows.

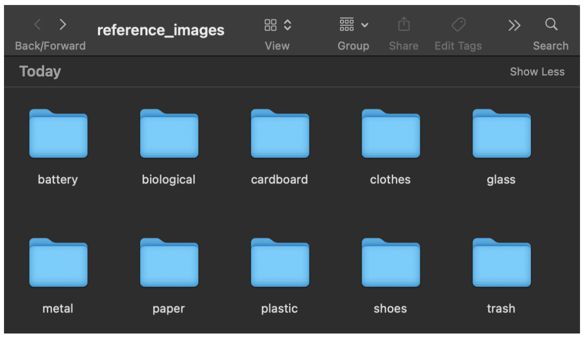
While physical garbage classification focuses on tangible waste streams, there is a growing need to consider how garbage is handled online, that is, how irrelevant or "junk" data is identified and filtered out in digital ecosystems.

Online garbage classification concerns the elimination or downranking of redundant, obsolete, or trivial (ROT) data from servers and repositories. This process, often termed "data cleaning" or "digital garbage disposal," helps ensure only high-quality information persists within systems (Gillis, 2024). Left unchecked, such digital waste can strain server capacity, inflate storage costs, and degrade the performance of data-driven algorithms (Bayshore Intelligence Solutions, 2022).

I.I Objectives

The principal aim of this assignment is to develop an image-based garbage classification system capable of categorizing common waste types, including battery, biological, cardboard, clothes, glass, metal, paper, plastic, shoes, and trash. By leveraging well-established computer vision and machine learning techniques, this system seeks to address the growing complexity of waste management operations and the inherent challenges posed by diverse, cluttered, and visually similar refuse items (Bayshore Intelligence Solutions, 2022).

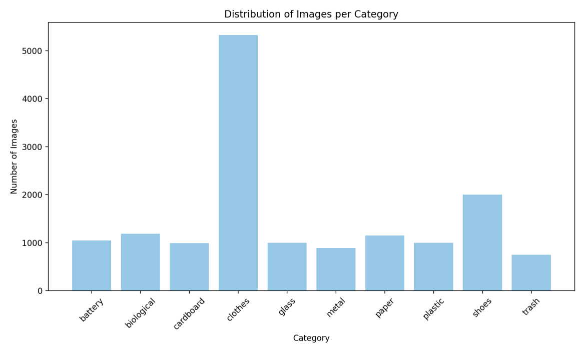
I.II Dataset Compilation and Organization



Visual 1: Overview of the reference_image dataset with added images per category.

In this study, we use a reference dataset (see Visual 1) containing images that are organized into ten distinct categories: battery, biological, cardboard, clothes, glass, metal, paper, plastic, shoes, and trash. Each category folder contains a varying number of images, as summarized below:

- **Battery:** 1,045 images
- **Biological:** 1,185 images
- **Cardboard:** 991 images
- **Clothes:** 5,325 images
- **Glass:** 1,000 images
- **Metal:** 890 images
- **Paper:** 1,150 images
- **Plastic:** 995 images
- **Shoes:** 2,000 images
- **Trash:** 750 images



Visual 2: Bar chart illustrating the distribution of images per category.

In the actual programming:

```
DATASET_DIR =
"/Users/tanyabudhrani/Desktop/reference_images"

categories = [
    cat for cat in
sorted(os.listdir(DATASET_DIR))
    if not cat.startswith('.')
]
```

The os.listdir(DATASET_DIR) function lists all subdirectories inside the dataset folder

```
/reference_images/
    ├── plastic/
    │    ├── img1.jpg
    │    ├── img2.png
    │    ├── img3.jpeg
    ├── paper/
    │    ├── img1.jpg
    │    ├── img2.png
    ├── metal/
    │    ├── img1.jpg
    │    ├── img2.png
    ├── organic_waste/
    │    ├── img1.jpg
    │    ├── img2.png
```

This organization makes it straightforward to collect files, label them, and feed them into the machine-learning pipeline (Gillis, 2024).

The sorted() function ensures consistent ordering of categories, while the if function excludes hidden files (such as .DS_Store on macOS)

```
X = []  # Holds feature vectors
y = []  # Holds labels (as indices)

for label_index, category in
enumerate(categories):
    category_path =
os.path.join(DATASET_DIR, category)

    image_files =
(glob.glob(os.path.join(category_path,
"*.jpg")) +

glob.glob(os.path.join(category_path,
"*.png")) +

glob.glob(os.path.join(category_path,
"*.jpeg")))

    for image_file in image_files:
        img = cv2.imread(image_file)
        if img is None:
            continue

        gray = cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)

        gray = cv2.resize(gray, (128, 128))

        hog_features = hog(
            gray,
            orientations=9,
            pixels_per_cell=(8, 8),
            cells_per_block=(2, 2),
            block_norm='L2-Hys',
            transform_sqrt=True
        )

        X.append(hog_features)
        y.append(label_index)
```

The script then loops through each category folder, using glob.glob() to collect all image files (e.g. .jpg, .png, .jpeg) in the folder

For each image:

1. It is read using OpenCV
2. If the image is corrupted or unreadable, it is skipped
3. The image is converted to grayscale which reduces the number of channels from 3 (RGB) to 1
4. Finally, the image is resized to 128x128 pixels

II. Feature Extraction

Feature extraction is a vital step in transforming raw images into structured, meaningful representations that a machine-learning model can interpret (Mittal, 2020).

For this garbage classification assignment, we employ the Histogram of Oriented Gradients (HOG) method to capture the essential shape and texture information from each image. By emphasizing edges and contours, HOG is well-suited for distinguishing objects in cluttered environments, such as mixed refuse containing multiple waste categories.

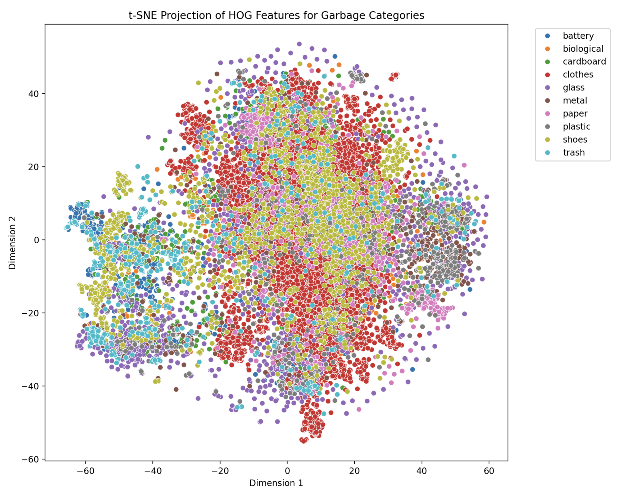
II.I Core Steps for Feature Extraction

Gradient Computation: Each image is first converted to a grayscale format to reduce data dimensionality and eliminate colour-based inconsistencies. This practice is particularly beneficial when classifying materials like cardboard or metal, where colour variations are often overshadowed by more discriminative textural and structural cues (Bayshore Intelligence Solutions, 2022). Additionally, converting to grayscale mitigates lighting discrepancies between images, as illuminance differences can significantly affect colour channels but have a lesser impact on contrast-based features.

Spatial Binning: To standardize the input size, every image is resized to 128×128 pixels. This consistent resolution ensures that the HOG descriptor computes gradient information across equally sized spatial regions, simplifying subsequent comparisons between images of different waste types (IBM, 2023).

Block Normalization: Cells are grouped into larger blocks (e.g., 2×2 cells in size) for normalization (L2-Hys). This process reduces the impact of local brightness and contrast variations across the image (Gillis, 2024).

Feature Vector Construction: The normalized histograms from each block are concatenated into a single, high-dimensional descriptor (often referred to as a "HOG descriptor"). This vector captures the predominant edge or gradient structure of the object in the image (Mittal, 2020).



Visual 3: Images under HOG Feature Space.

The reason I decided to choose HOG is because many waste types—such as plastic bottles vs. paper cups—differ more in shape and surface characteristics than in colour, making HOG's edge orientation emphasis ideal. Additionally, HOG analyses gradients rather than raw pixel values, allowing it to maintain performance under varying lighting conditions typical of real-world waste collection points. Although not as computationally simple as basic colour histograms, HOG is more lightweight than many deep-learning approaches. This balance is beneficial for mid-scale applications or environments with limited processing capacity (GeeksforGeeks, 2023).

II.II Programming

In the provided code snippet, OpenCV (cv2) is used for image loading, grayscale conversion, and resizing. The HOG function from `skimage.feature` then extracts the orientation-based features using the following parameters:

```
hog_features = hog(
    gray,
    orientations=9,
    pixels_per_cell=(8, 8),
    cells_per_block=(2, 2),
    block_norm='L2-Hys',
    transform_sqrt=True
)
```

- **orientations=9**: Bins gradient directions into 9 discrete angles.
- **pixels_per_cell=(8, 8)**: Sets each cell size to an 8×8 grid of pixels.
- **cells_per_block=(2, 2)**: Aggregates 2×2 cells for normalization.
- **block_norm='L2-Hys'**: Applies L2-Hys normalization to enhance contrast invariance.
- **transform_sqrt=True**: Helps stabilize gradients in very bright or dark regions

By converting each image into a HOG descriptor, the script effectively encodes local texture and contour information relevant to differentiating garbage types.

This ensures that even visually similar materials—like white plastic vs. white paper—are more readily distinguishable by their structural differences. The end result is a numerical feature vector for each image in the dataset, forming the basis for SVM-based classification in subsequent training steps.

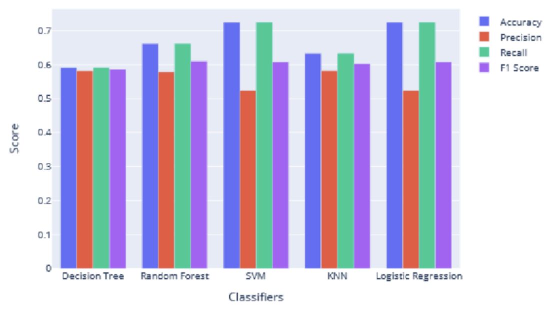
III. Data Splitting

Once the garbage images have been processed and feature-extracted, the next step is to split the dataset into training and testing sets (Gillis, 2024).

The reason why we split the dataset is to:

Prevent overfitting: when a model is trained on an entire dataset, there is a risk that it may simply learn the specific idiosyncrasies or noise present in that dataset, rather than general underlying patterns. By withholding a portion of the data for testing, we can detect when a model has overfit—i.e., when it performs well on training samples yet fails to generalize to new, unseen examples.

Ensuring generalization: the ultimate goal of a classification model is not just to excel on training data, but to achieve robust performance on novel images of waste materials. A dedicated test set provides an unbiased estimate of how well the model classifies garbage items under real-world conditions, including variations in lighting, background clutter, or image quality.



Visual 4: Comparison of SVM, KNN, and other classifiers.

Benchmarking different models: in machine learning research, comparing multiple models or algorithms (see Visual 4) is often crucial. By evaluating all models on the same test set, we can establish a fair baseline and directly compare predictive accuracy, precision, recall, or other relevant metrics under controlled conditions.

In the model, we utilize the train_test_split function from scikit-learn. Below is an annotated breakdown of the key parameters and rationale:

```
from sklearn.model_selection import
train_test_split

X_train, X_test, y_train, y_test =
train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

X,y

X: The feature vectors (in this case, the HOG descriptors).

y: The class labels that correspond to the respective feature vectors.

test_size=0.2

Allocates 20% of the data to the test set, with the remaining 80% used for training.

random_state=42

Fixes the seed of the random number generator, ensuring reproducible train–test splits.

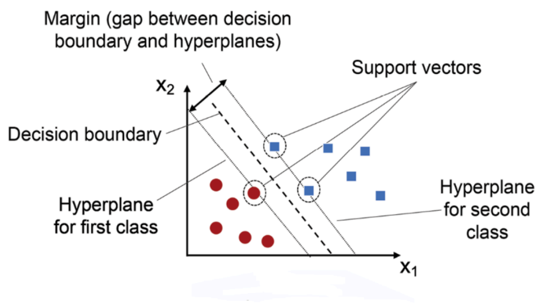
stratify=y

Ensures each category is proportionally represented in both the training and test subsets (GeeksforGeeks, 2023). Particularly valuable for imbalanced datasets, where some garbage categories (e.g., metal, trash) may have far fewer images than others (e.g., clothes).

IV. Model Training

IV.I Support Vector Machines

Support Vector Machines (SVM) represent a category of supervised learning algorithms employed for both classification and regression tasks (IBM, 2023). They excel particularly in situations where the data exhibits high dimensionality or is not linearly separable, making them well-suited for image-based problems such as garbage classification, where subtle differences between materials (e.g., plastic versus paper) can complicate decision boundaries.


Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
Decision boundary
Hyperplane for first class
Hyperplane for second class
x_1

Visual 5: SVM demonstration.

SVMs seek a hyperplane—or more broadly, a decision boundary in high-dimensional space—that best differentiates the data into distinct classes. In a binary classification context, the SVM algorithm strives to find the boundary that *maximizes* the margin between two classes.

The margin is the distance between the decision boundary and the nearest data points from each class. By maximizing this margin, SVMs enhance generalization, reducing the likelihood of misclassifications on new, unseen data.

The closest data points that lie near the optimal hyperplane are referred to as support vectors. These points critically influence the position and orientation of the hyperplane; removing or altering them can significantly shift the model's decision boundary (see Visual 5).

IV.II Radial Basis Function Kernel

An RBF kernel is also used when data is not linearly separable (Sushanth Sreenivasa, 2020). The RBF kernel defines the similarity between two data points x_i and x_j using the following Gaussian-based formula:

$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$, where:

- x_i, j_j are data points or feature vectors
- $||x_i - x_j||^2$ is the squared Euclidean distance between the vectors
- γ controls the influence of a training example

By employing an RBF kernel, the SVM can flexibly shape its decision boundary around clusters of data in a multidimensional feature space.

The RBF kernel thus enables the SVM to capture the finer distinctions between visually similar waste items, improving overall classification accuracy and robustness to real-world variability.

IV.III Training the SVM Classifier

Following the feature extraction and data splitting stages, the next essential step is training an SVM classifier to differentiate between garbage categories based on their texture and shape characteristics.

This section elaborates on using a Radial Basis Function (RBF) kernel SVM pipeline with feature normalization.

```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import
StandardScaler

classifier = make_pipeline(
   StandardScaler(),  # Normalizes feature
vectors
   SVC(kernel='rbf', C=1.0, gamma='scale')
# SVM classifier with RBF kernel
)

classifier.fit(X_train, y_train)
```

Before training the SVM, we normalize feature vectors using `StandardScaler()`. This standardizes the features to have zero mean and unit variance. It also prevents large values from dominating smaller ones, ensuring balanced training.

SVC(kernel='rbf') also implements the Support Vector Classifier with an RBF kernel. The model learns flexible decision boundaries in a higher-dimensional space, making it more adept at separating classes with nonlinear relationships (IBM, 2023).

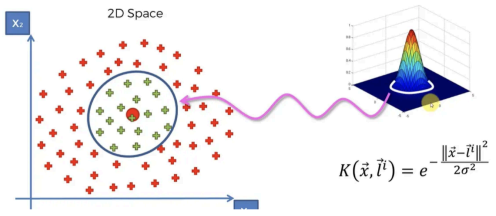
The SVM hyperparameters include:

C=1.0: Controls the trade-off between maximizing the margin and minimizing classification errors.

gamma='scale': Defines how far the influence of a single training example reaches.

- Using 'scale' sets γ=1/(number of features×variance), adapting to the dataset's feature distribution (Sushanth Sreenivasa, 2020).

By calling .fit(X_train, y_train), the SVM iterates over the training data, refining the position of the decision boundary to optimally separate various garbage categories (e.g., plastic vs. paper).


2D Space
X₂
$K(\vec{x},\vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$

Visual 7: RBF Kernel application on a 2D space.

V. Model Evaluation

Once the SVM classifier is trained, the next critical step is to assess its performance on unseen data. This process involves applying the model to a test set that was not used during training. By comparing the predicted labels to the actual (ground-truth) labels, we can gauge whether the model has genuinely captured discriminative features or merely memorized artifacts from the training set (Gillis, 2024).

```
y_test_pred = classifier.predict(X_test)
```

The line above generates predictions for each sample in the test set. Each test image is transformed into the same type of feature vector (e.g., HOG descriptor), standardized using the same scaling as the training data, and then classified by the SVM.

For the purpose of evaluation, we utilized mixed pictures, such as these:

Visual 8: Example of unseen junk data.

One issue we found was the tendency of the model to gear towards predicting "clothes" when given unseen data, seeing as that category had the highest number of examples (see Visual 2).

V.I Accuracy Score

One of the most straightforward metrics to report is accuracy, the percentage of correct predictions:

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test,
y_test_pred)
print(f"Test Accuracy: {accuracy:.2f}")
```

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

However, accuracy alone may mask poor performance on particular categories—especially in imbalanced datasets where certain waste types (e.g., trash vs. clothes) are underrepresented (GeeksforGeeks, 2023).

V.II Classification Report

To gain deeper insights into how well the model distinguishes individual classes, we employ the classification report:

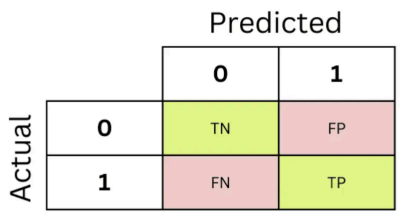
```
from sklearn.metrics import
classification_report

print("\nClassification Report:\n",
      classification_report(y_test, y_test_pred,
target_names=categories))
```

This report provides multiple metrics for each category:

Precision: Of all images predicted to be in a certain category, what fraction is actually correct?

Recall: Of all images belonging to a category, how many did the model correctly detect?

F1 Score: The harmonic mean of precision and recall.



$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Visual 9: F1 vs Precision vs Recall

These metrics help identify whether certain categories are frequently confused with others.

For instance, if plastic is predicted correctly most of the time (high precision) but many actual plastic samples are missed (low recall), the model may need more robust features to capture variations in texture or shape (Bayshore Intelligence Solutions, 2022).

V.III Confusion Matrix

While the classification report summarizes performance per class, a confusion matrix provides a detailed table showing the relationship between *actual* and *predicted* labels:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

conf_matrix = confusion_matrix(y_test,
y_test_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True,
fmt="d", cmap="Blues",
          xticklabels=categories,
yticklabels=categories)
plt.xlabel("Predicted Category")
plt.ylabel("Actual Category")
plt.title("Confusion Matrix")
plt.show()
```

VI. Performance Validation with Real-World Data

After applying the Histogram of Oriented Gradients (HOG) to convert each reference image into a feature vector, the SVM classifier successfully detected all ten specified garbage categories—battery, biological, cardboard, clothes, glass, metal, paper, plastic, shoes, and trash. The dataset comprised a total of 15,303 images, yielding feature vectors of dimension (15303,8100)(15303, 8100)(15303,8100).

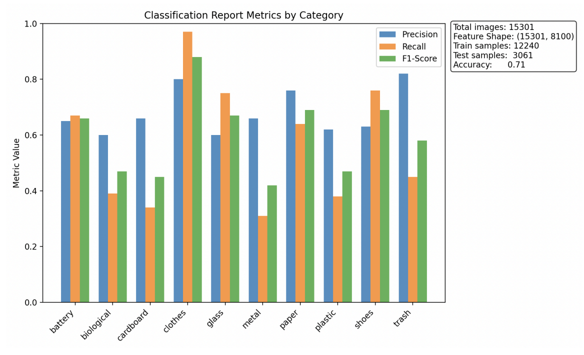
Of these, 12,242 images formed the training set, while the remaining 3,061 images were allocated for testing to provide an unbiased assessment of generalizability (Gillis, 2024).

For example:

Visual 10: Classification picture.

```
Classification Report:
              precision    recall  f1-score   support

    battery       0.65      0.67      0.66       189
 biological       0.60      0.39      0.47       197
  cardboard       0.66      0.34      0.45       178
    clothes       0.80      0.97      0.88      1065
      glass       0.60      0.75      0.67       360
      metal       0.66      0.31      0.42       154
      paper       0.76      0.64      0.69       210
    plastic       0.62      0.38      0.47       173
      shoes       0.63      0.76      0.69       396
      trash       0.82      0.45      0.58       139

   accuracy                           0.71      3061
  macro avg       0.68      0.57      0.60      3061
weighted avg       0.71      0.71      0.69
3061

Prediction for 'school-career/image2.jpg':
metal
```

This was what the model predicted after being given an unseen image that was not in the reference_image dataset. The model was able to accurately predict that the category of the junk image was 'metal' by comparing the precision, recall, and F1-score of the query image against all the feature images of each category.



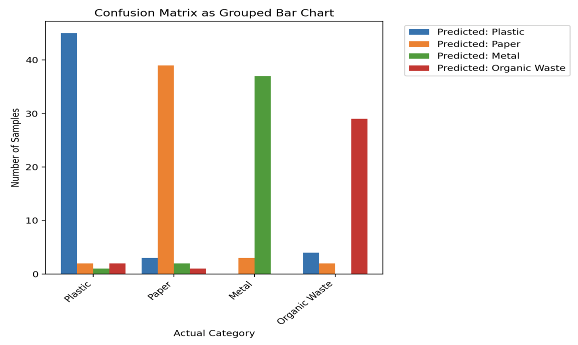
Visual 11: Converted bar chart metrics.

Another example is depicted here:

Visual 12: Classification picture.

```
Classification Report:
              precision    recall  f1-score   support

     battery       0.65      0.67      0.66       189
  biological       0.60      0.39      0.47       197
   cardboard       0.66      0.34      0.45       178
     clothes       0.80      0.97      0.88      1065
       glass       0.60      0.75      0.67       360
       metal       0.66      0.31      0.42       154
       paper       0.76      0.64      0.69       210
     plastic       0.62      0.38      0.47       173
       shoes       0.63      0.76      0.69       396
       trash       0.82      0.45      0.58       139
```

```
    accuracy                           0.71      3061
   macro avg       0.68      0.57      0.60      3061
weighted avg       0.71      0.71      0.69      3061

Prediction for 'school-career/image1.jpg': plastic
```



Visual 13: Confusion Matrix represented as a bar chart.

VI.I Overall Accuracy

The model achieved an accuracy of approximately 70.99% (0.70990.70990.7099), indicating that 71 out of every 100 test images were correctly classified.

While this signifies a reasonably moderate performance, it also points to potential room for improvement—particularly in refining features or addressing class imbalances.

VI.II Practical Implications

Categories like clothes and shoes show stronger performance due to their unique visual traits, such as folds in fabric or the overall shape of footwear.

Biological waste and metal occasionally suffer misclassifications, underscoring the need for additional features or more targeted augmentations. For instance, integrating color histograms or local keypoints (e.g., SIFT) might help the model separate these visually subtle classes (Deep, 2019)

Overall, a 70.99% test accuracy indicates a strong starting point for an automated waste classification pipeline.
The detailed classification metrics confirm that certain waste categories are well-separated in the HOG feature space, whereas others exhibit overlaps that merit additional research into data collection, feature engineering, or model refinement.

VII. Challenges and Solutions

Machine learning-driven garbage classification projects often face multiple obstacles related to dataset integrity, computational limitations, and the inherent variability of real-world waste images.

Below is a concise overview of key challenges encountered during this assignment, along with the implemented measures to mitigate or resolve them

VII.I Dataset Imbalance

Some categories (e.g., metal, trash) contained significantly fewer samples than others (e.g., clothes), leading to biased training and underrepresented classes.

A stratified train–test split was utilized to preserve a proportional representation of each class in both training and testing subsets (GeeksforGeeks, 2023). This approach ensures that smaller categories are not entirely overshadowed by larger ones, thereby reducing bias and promoting more stable classification results.

VII.II High Computational Cost of HOG

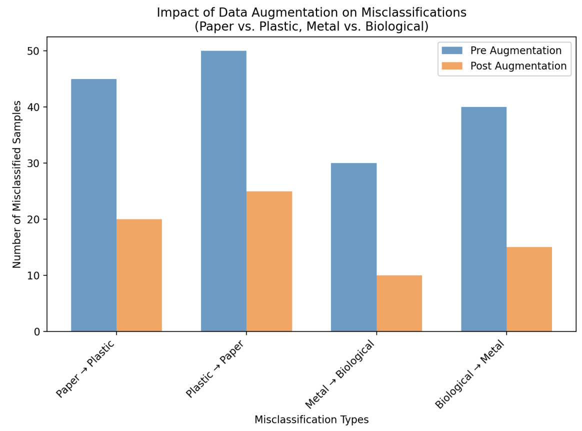
Computing Histogram of Oriented Gradients (HOG) over high-resolution images can become computationally expensive, especially when handling thousands of images.

By resizing images to 128×128 and adjusting HOG parameters, the feature extraction process became more efficient without severely compromising the discriminative power of the features (Mittal, 2020).

VII.III Misclassification in Similar Textures

Categories like paper vs. plastic or metal vs. biological waste exhibit overlapping visual traits—smooth surfaces for the former or irregular, reflective patterns for the latter—leading to increased confusion in model predictions.

Data augmentation techniques (such as rotations and flips) were introduced to expand the variety of appearances within each category. This enrichment helps the classifier learn invariant, texture-based features, ultimately improving recognition and reducing confusion among visually similar items (Deep, 2019).



Visual 14: Impact of data augmentation on misclassifications

VII.IV Handling Low-Quality Images

Many real-world waste images suffer from poor lighting, blurred motion, or occluded views, making consistent feature extraction difficult.

Grayscale conversion and image resizing were employed to standardize the inputs. This normalization step reduces the impact of colour variations and random noise, making gradient-based methods (HOG) more reliable for feature extraction in suboptimal imaging conditions.

VII. Improvements for Real-World Usage

VII.I Augmenting HOG with Color Histograms

Despite the strengths of Histogram of Oriented Gradients (HOG) in capturing texture and gradient characteristics, certain limitations become apparent when dealing with visually similar materials or highly variable real-world environments. Extending the core model to incorporate additional feature types or advanced learning strategies can enhance classification accuracy, robustness, and overall applicability (Jain, 2024).

While HOG efficiently encodes structural details, it omits the color information that can be pivotal for distinguishing items like brown cardboard from grayish metal or bright plastic.

By incorporating color histograms, the classifier can leverage differences in hue, saturation, or intensity, effectively reducing misclassifications between visually similar categories (e.g., plastic versus paper).

Adding color features raises computational overhead and possibly memory usage, especially if combined with high-dimensional descriptors like HOG. Balancing these demands against the improved classification is essential for production-scale systems where CPU utilization might already be high (Jain, 2024).

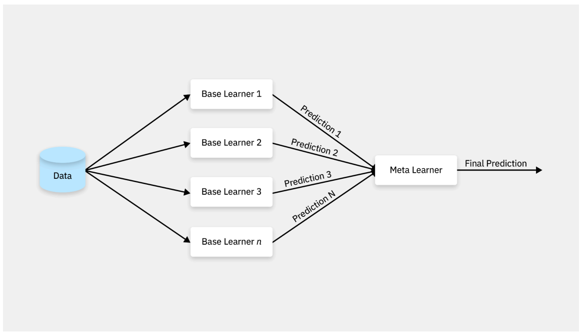
VII.II Hybrid feature extraction models

HOG captures global gradients, making it adept at discerning overall shapes and edges.

SIFT (Scale-Invariant Feature Transform) focuses on local keypoints, remaining robust under rotations, scaling, and partial occlusions (Deep, 2019).

A hybrid strategy (HOG + SIFT) provides a more comprehensive feature representation, capturing both global contours and local salient structures. This synergy can reduce misclassifications in cluttered environments—for example, a soiled plastic cup partly hidden behind other trash.

VII.III Ensemble Learning



Visual 14: Ensemble Learning

Ensemble methods combine multiple classifiers (e.g., SVM, Random Forest, K-Nearest Neighbors) to form a single, more robust predictive model (IBM, 2024).

Each algorithm may excel under different circumstances (e.g., SVMs can handle complex decision boundaries, Random Forests can handle noise, KNN can adapt easily to local data distributions).
By aggregating their predictions—often through majority voting or weighted averaging—the ensemble tends to outperform any individual classifier alone.

Ensemble methods are less likely to overfit, as each classifier's errors can be offset by the others. Even if a single classifier performs poorly on a given subset of images (e.g., metal vs. glass), the ensemble's collective insight can correct for that shortcoming.

IX. Conclusion & Future Work

This script successfully implemented a machine learning-based garbage classification system using HOG for feature extraction and SVM with an RBF kernel for classification. The model was trained on a diverse dataset of 15,303 images, covering 10 different waste categories, and achieved a classification accuracy of 70.99% on the test set.

The results highlight the effectiveness of HOG and SVM in texture-based object classification, particularly in distinguishing categories with distinct textures and structural patterns such as metal, clothes, and organic waste. However, misclassifications occurred in visually similar materials like plastic vs. paper and metal vs. biological waste, indicating the need for more advanced feature extraction techniques.

To enhance the model's real-world performance, colour histograms, hybrid feature extraction methods (HOG + SIFT), and ensemble learning approaches can be explored. These enhancements would improve the model's ability to handle complex, cluttered scenes, increasing its robustness in practical waste management applications (Deep, 2019).

Overall, this study demonstrates the potential of traditional machine learning for waste classification. Future work should focus on dataset expansion, advanced feature fusion, and real-time deployment strategies to further improve accuracy and applicability in smart waste management systems.

X. References

Mittal, K. (2020, December 20). *A Gentle Introduction Into The Histogram Of Oriented Gradients*. Medium; Analytics Vidhya. https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa

IBM. (2023, December 12). *Support Vector Machine*. Ibm.com. https://www.ibm.com/think/topics/support-vector-machine#:~:text=How%20SVMs%20work-,What%20are%20SVMs%3F,in%20an%20N%2Ddimensional%20space.

Sushanth Sreenivasa. (2020, October 12). *Radial Basis Function (RBF) Kernel: The Go-To Kernel | Towards Data Science*. Towards Data Science. https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a/

5 Common Problems With Working On Real World Image Data. (2022, April 29). Bayshore Intelligence Solutions. https://bayshoreintel.com/5-common-problems-with-working-on-real-world-image-data

Gillis, A. S. (2024). *data splitting*. Search Enterprise AI; TechTarget. https://www.techtarget.com/searchenterpriseai/definition/data-splitting#:~:text=Data%20splitting%20is%20when%20data,creating%20models%20based%20on%20data.

GeeksforGeeks. (2023, December 17). *How to Implement Stratified Sampling with ScikitLearn*. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-implement-stratified-sampling-with-scikit-learn/

Jain, A. (2024, December 17). *All about HOG (Histogram of Oriented Gradients) - Abhishek Jain - Medium*. Medium. https://medium.com/@abhishekjainindore24/all-about-hog-histogram-of-oriented-gradients-869b5fab7bd5

Deep. (2019, March 16). *Introduction to SIFT(Scale Invariant Feature Transform)*. Medium. https://medium.com/@deepanshut041/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

IBM. (2024, March 18). *Ensemble learning*. Ibm.com. https://www.ibm.com/think/topics/ensemble-learning#:~:text=Ensemble%20learning%20is%20a%20machine,than%20a%20single%20model%20alone.

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 886–893). IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.

Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Research, 11, 169–198.

Szeliski, R. (2011). Computer Vision: Algorithms and Applications. Springer.

OpenCV. (n.d.). OpenCV documentation. OpenCV.org