# Comprehensive Technical Report

## Automated Material Stream Identification System

Github Repo URL: https://github.com/NourEldeenM/material-stream-identification-system/tree/main

## 1. Summary

This report presents a comprehensive summary of an Automated Material Stream Identification (MSI) system designed to classify waste materials into six categories: Glass, Paper, Cardboard, Plastic, Metal, and Trash. The system achieves **95.33% accuracy** using DL-based feature extraction (ResNet50) combined with Support Vector Machine (SVM) classification.

**Key Points:**

- Implemented data augmentation increasing dataset size (1,960 → 3,000 image)
- Feature Extraction using ResNet50 CNN
- KNN and SVM classifiers
- Developed real-time classification system with web-based interface

---

## 2. Feature Extraction Methods

**ResNet50 Convolutional Neural Network**

ResNet50 is a pre-trained DL model with 50 layers.

**Overview:**

- **Depth:** 50 layers
- **Pre-training:** ImageNet dataset
- **Feature Extraction:** Average Pooling
- **Output dimensions:** 2,048 features

**Implementation :**

```
ResNet50(weights='imagenet', include_top=False, pooling='avg')
```

- Remove final classification layer (include_top=False)
- Apply average pooling
- Extract 2,048-dimensional features
- Input image size: 224×224×3

**Results:**

- Training Accuracy: 100%

- Testing Accuracy: **95.33%** on SVM classifier and **93.17%** on KNN classifier

**Advantages:**

1. **Semantic features**
2. **Transfer learning**
3. **Reduced dimensionality**
4. **Better generalization**
5. **Robust to variations**

---

# 3. Classifiers Comparison

## 3.1 k-Nearest Neighbors (k-NN)

**Algorithm Overview**

k-NN is a non-parametric, instance-based learning algorithm that classifies samples based on the majority vote of their k nearest neighbors.

**Classification Decision:**

```
For a new sample:
1. Calculate distance to all training samples
2. Select k nearest neighbors
3. Majority vote determines class
```

**Implementation Details**

**Hyperparameters:**

- **k (neighbors):** 3 (optimized via GridSearchCV)
- **Distance metric:** Manhattan (optimized via GridSearchCV)
- **Weighting:** Distance-weighted (closer neighbors have more influence) (optimized via GridSearchCV)
- **Rejection threshold:** 0.6 (confidence-based unknown detection)

**Distance Metrics Evaluated:**

- **Euclidean:** $\sqrt{(\Sigma(xi - yi)^2)}$
- **Manhattan:** $\Sigma|xi - yi|$
- **Minkowski:** $(\Sigma|xi - yi|^p)^{(1/p)}$

**Feature Preprocessing**

**StandardScaler Normalization:**

```
X_scaled = (X - mean) / std_deviation
```

**Hyperparameter Tuning**

**Grid Search:**

```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 15],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}
```

**Cross-Validation:**

- 5-fold stratified cross-validation

**Optimal Parameters Found:**

- k = 3
- weights = 'distance'
- metric = 'euclidean'

**Rejection Mechanism**

**Confidence-Based Unknown Detection:**

```
if max_probability < rejection_threshold:
    prediction = 'Unknown'
else:
    prediction = argmax(probabilities)
```

**Performance Results**

| Metric | Value |
|---|---|
| Training Accuracy | 100% |
| Testing Accuracy | 93.17% |
| Best k | 3 |
| Best Metric | Euclidean |
| Feature Dimensions | 2,048 |
| Training Samples | 2,400 |
| Testing Samples | 600 |

---

## 3.2 Support Vector Machine (SVM)

**Algorithm Overview**

SVM finds the optimal hyperplane that maximizes the margin between classes in high-dimensional space.

**Objective:** Maximize margin while minimizing classification error

**Implementation Details**

**Multi-class Strategy:** One-vs-Rest (OvR):

- Train 6 binary classifiers (one per class)
- Each classifier: "class i" vs. "all other classes"
- Final prediction: class with the highest confidence

**Probability:**

```
probability=True
```

Converts SVM decision scores to probabilities for confidence estimation.

**Performance Results**

| Metric | Value |
|---|---|
| Training Accuracy | 100% |
| Testing Accuracy | **95.33%** |
| Kernel | rbf |
| C | 3 |
| Gamma | scale |

---

# 4. Data Augmentation

**Initial Dataset:**

- Total: 1,960 images
- 95 Corrupt images

**Target:**

- Minimum 30% increase
- Balanced classes
- Final: 3,000 images

## Augmentation Techniques

1. **Rotation (±30°)**

2. **Horizontal Flip**
3. **Vertical Flip**
4. **Brightness Up (1.1-1.3×)**
5. **Brightness Down (0.7-0.9×)**
6. **Scale Up (1.1-1.2×)**
7. **Scale Down (0.8-0.9×)**

### Implementation Strategy

- Skipped 95 corrupt images
- Validated all augmented images
- Maintained original test set separate (to prevent data leakage)

---

## 5. System Architecture

### Overview

```
Camera → Image Capture → Preprocessing → Feature Extraction → Classification → Display
         (224×224 RGB)   (Normalize)    (ResNet50)         (SVM/KNN)          (UI)
```

### Tech Stack

**Backend:**

- FastAPI (REST API)
- Python
- TensorFlow/Keras (ResNet50)
- scikit-learn (KNN/SVM)
- OpenCV

**Frontend:**

- React.js
- Vite build tool
- Webcam component

---

## Models Configuration Details

### k-NN Classifier Configuration

```
KNeighborsClassifier(
    n_neighbors=3,
    weights='distance',
    metric='manhattan',
```

```
    n_jobs=-1
)
```

### SVM Classifier Configuration

```
SVC(
    kernel='rbf',
    C=30,
    gamma='scale',
    rejection_threshold=0.4
)
```

### ResNet50 Feature Extractor Configuration

```
ResNet50(
    weights='imagenet',
    include_top=False,
    pooling='avg'
)
```