

# Technical Report: Material Stream Identification System

---

Cairo University - Faculty of Engineering

---

Machine Learning Project - Fall 2025

mazen wael mahmoud 20226084  
mohamed essam 20226093  
mohamed mohsen 20226094  
abdelrahman amer 20226056  
mahmoud mohamed 20226097

## Table of Contents

---

1. [Introduction](#)
2. [Data Preprocessing and Augmentation](#)
3. [Feature Extraction Methodology](#)
4. [Classifier Implementation](#)
5. [Unknown Class Rejection Mechanism](#)
6. [Results and Performance Analysis](#)
7. [Architecture Comparison: SVM vs k-NN](#)
8. [Real-Time System Deployment](#)
9. [Real-World Testing Results](#)
10. [Conclusion](#)

## 1. Introduction

---

### 1.1 Project Overview

This project implements an Automated Material Stream Identification (MSI) System for waste classification using fundamental Machine Learning techniques. The system classifies waste materials into seven categories: Glass, Paper, Cardboard, Plastic, Metal, Trash, and Unknown.

## 1.2 Objectives

- Develop a robust feature extraction pipeline converting raw images to numerical vectors
- Implement and compare SVM and k-NN classifiers
- Achieve  $\geq 85\%$  validation accuracy on six primary material classes
- Deploy a real-time classification system using live camera feed

## 1.3 Material Classes

ID	Class	Description
0	Cardboard	Heavy-duty structured cellulose fiber material
1	Glass	Amorphous solid materials, primarily silicates
2	Metal	Elemental or compound metallic substances
3	Paper	Thin materials from pressed cellulose pulp
4	Plastic	High-molecular-weight organic compounds
5	Trash	Non-recyclable or contaminated waste
6	Unknown	Out-of-distribution items or blurred inputs

## 2. Data Preprocessing and Augmentation

---

### 2.1 Original Dataset

- **Total Images:** 2,527 images across 6 classes
- **Class Distribution:** Imbalanced (ranging from ~400 to ~500 per class)
- **Original Image Sizes:** Variable dimensions

### 2.2 Image Preprocessing

All images undergo standardized preprocessing:

1. **Resizing:** All images resized to 96×96 pixels - Balances computational efficiency with feature quality - Consistent input size for feature extraction
2. **Color Space:** Preserved BGR format for multi-channel feature extraction

## 2.3 Data Augmentation Techniques

We applied data augmentation to increase training data by **~1100%** (exceeding the 30% minimum requirement):

Technique	Description	Justification
Horizontal Flip	Mirror image horizontally	Waste items can appear in any orientation
Vertical Flip	Mirror image vertically	Simulates different viewing angles
Rotation (90°, 180°)	Rotate by fixed angles	Objects on conveyor belts rotate
Brightness Adjustment ( $\pm 30$ )	Increase/decrease brightness	Handles varying lighting conditions
Contrast Adjustment (0.7x, 1.3x)	Modify image contrast	Adapts to different camera settings
Gaussian Noise	Add random noise ( $\sigma=15$ )	Improves robustness to sensor noise
Gaussian Blur (3×3)	Slight blur	Handles out-of-focus images

## 2.4 Augmentation Results

- **Original Dataset:** 2,527 images
- **Augmented Dataset:** 30,000 images (5,000 per class)
- **Increase:** ~1,087% (far exceeding 30% requirement)
- **Class Balance:** Perfectly balanced (5,000 images each)

## 3. Feature Extraction Methodology

---

## 3.1 Overview

We convert each  $96 \times 96 \times 3$  image into a **187-dimensional feature vector**. This fixed-length numerical representation captures color, texture, shape, and gradient information essential for material classification.

## 3.2 Feature Categories

### 3.2.1 Color Statistics (45 features)

**BGR Channel Statistics (21 features)** - Mean, standard deviation, median, 25th/75th percentiles, min, max for each channel - **Justification:** Different materials have distinct color profiles (metal is gray, cardboard is brown)

**HSV Channel Statistics (15 features)** - Mean, standard deviation, median, 25th/75th percentiles for H, S, V channels - **Justification:** HSV separates color (hue) from intensity, important for material identification under varying lighting

**LAB Channel Statistics (9 features)** - Mean, standard deviation, median for L, A, B channels - **Justification:** LAB is perceptually uniform; L channel isolates lightness for reflectivity detection

### 3.2.2 Grayscale Statistics (8 features)

- Mean, std, variance, min, max, range, 25th/75th percentiles
- **Justification:** Captures overall brightness distribution for material type differentiation

### 3.2.3 Texture Features (27 features)

**Laplacian Texture (3 features)** - Variance, mean absolute value, standard deviation of Laplacian - **Justification:** Measures edge content; smooth materials (plastic) differ from textured (paper)

**Sobel Gradient (8 features)** - Gradient magnitude statistics (mean, std, max, 90th percentile) - Gradient direction statistics (mean, std) - Separate X and Y gradient means - **Justification:** Captures edge orientation patterns characteristic of different materials

**LBP-like Texture (16 features)** - Local Binary Pattern histogram (16 bins) - Computed by comparing each pixel to 4 neighbors (up, right, down, left) - **Justification:** Captures micro-texture patterns; paper has fine texture, glass is smooth

### 3.2.4 Edge Features (4 features)

- Canny edge density (ratio of edge pixels)
- Mean and std of edge map
- Normalized edge sum
- **Justification:** Metal and glass have sharp edges; cardboard has irregular edges

### 3.2.5 Color Histogram Features (48 features)

**BGR Histograms (24 features)** - 8 bins per channel, normalized - **Justification:** Captures color distribution independent of spatial arrangement

**HSV Histograms (24 features)** - 8 bins per channel, normalized - **Justification:** Hue histogram distinguishes material colors; saturation indicates colorfulness

### 3.2.6 Reflectivity Features (5 features)

- Brightness variance
- Local contrast ratio
- Bright spot ratio (>95th percentile)
- Dark spot ratio (<5th percentile)
- Dynamic range (max - min)
- **Justification:** Metal and glass are highly reflective; paper and cardboard are matte

### 3.2.7 HOG-like Gradient Histogram (36 features)

- Image divided into  $2 \times 2$  grid (4 cells)
- 9 orientation bins per cell
- Normalized per cell
- **Justification:** Captures shape and structure; bottles have curved edges, boxes have straight edges

### 3.2.8 Additional Texture Features (8 features)

- Gradient magnitude percentiles (10th, 30th, 50th, 70th)
- Laplacian statistics (mean, 25th/75th percentiles, coefficient of variation)
- **Justification:** Fine-grained texture characterization for distinguishing similar materials

### 3.2.9 Color Coherence Features (6 features)

- Coherence ratio for each BGR channel (pixels within  $\pm 30$  of mean)
- Coherence ratio for each HSV channel

- **Justification:** Uniform materials (plastic) have high coherence; mixed materials (trash) have low coherence

### 3.3 Feature Vector Summary

Category	Feature Count	Purpose
Color Statistics	45	Material color properties
Grayscale Statistics	8	Brightness distribution
Texture Features	27	Surface texture patterns
Edge Features	4	Edge characteristics
Color Histograms	48	Color distribution
Reflectivity	5	Surface reflectance
HOG-like	36	Shape and structure
Additional Texture	8	Fine texture details
Color Coherence	6	Color uniformity
<b>Total</b>	<b>187</b>	

### 3.4 Feature Normalization

All features are normalized using **StandardScaler** (z-score normalization):

- Centers each feature to zero mean
- Scales to unit variance
- **Justification:** Essential for SVM with RBF kernel; prevents features with larger ranges from dominating

## 4. Classifier Implementation

---

### 4.1 Support Vector Machine (SVM)

### 4.1.1 Architecture

- **Kernel:** Radial Basis Function (RBF)
- **Parameters:**
  - C = 0.5 (regularization parameter)
  - gamma = 0.01 (kernel coefficient)

### 4.1.2 Kernel Choice Justification

We chose the RBF kernel because: 1. **Non-linear Decision Boundaries:** Material classification requires complex, non-linear boundaries in feature space 2. **Flexibility:** RBF can model any smooth decision boundary 3. **Single Parameter:** Only gamma needs tuning (compared to polynomial kernel's degree) 4. **Universal Approximator:** Can approximate any continuous function given enough support vectors

### 4.1.3 Parameter Selection

- **C = 0.5:** Moderate regularization prevents overfitting while allowing flexible boundaries
  - Lower C (0.5) vs typical (1.0) reduces overfitting on augmented data
- **gamma = 0.01:** Controls influence radius of training examples
  - Low gamma creates smoother boundaries, better generalization
  - Chosen via cross-validation from {0.001, 0.01, 0.1, 1.0}

### 4.1.4 Multi-class Strategy

- **One-vs-One (OvO):** Default for sklearn SVC with RBF
- Creates  $C(6,2) = 15$  binary classifiers
- Final prediction by majority voting

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

### 4.2.1 Architecture

- **k (neighbors):** 6
- **Weights:** Distance-based
- **Distance Metric:** Manhattan (L1)

### 4.2.2 Parameter Justification

**k = 6:** - Odd number preferred but 6 chosen based on cross-validation - Balances between: - Too small k (noise sensitivity) - Too large k (over-smoothing class boundaries) - 6 neighbors provides stable voting for 6-class problem

**Distance Weighting:** - Closer neighbors contribute more to the prediction - Weight = 1/distance -

**Justification:** Reduces influence of distant neighbors that may be from different classes

**Manhattan Distance:** - L1 norm:  $\sum |x_i - y_i|$  - **Justification:** - More robust to outliers than Euclidean - Works well in high-dimensional spaces (187 features) - Less sensitive to irrelevant features - Cross-validation showed better accuracy than Euclidean

#### 4.2.3 Computational Considerations

- No explicit training phase (lazy learner)
- All computation at prediction time
- Feature scaling critical for distance-based methods

## 5. Unknown Class Rejection Mechanism

---

### 5.1 Overview

The system implements a dual rejection mechanism to handle out-of-distribution samples and poor-quality inputs.

### 5.2 Confidence-Based Rejection

- **Threshold:** 0.30 (30% minimum confidence)
- Predictions with confidence below threshold are rejected as "Unknown"
- **SVM Confidence:** Derived from decision function margins using sigmoid transformation
- **k-NN Confidence:** Based on neighbor voting ratio and inverse distance weighting

### 5.3 Blur Detection

- **Method:** Laplacian variance
- **Threshold:** 50.0
- Low variance indicates lack of edges (blurry image)
- Blurry images are classified as "Unknown" regardless of model prediction

## 5.4 Implementation

```
class UnknownClassHandler:  
    UNKNOWN_CLASS_ID = 6  
  
    def apply_rejection(self, prediction, confidence, is_blurry):  
        if is_blurry or confidence < self.confidence_threshold:  
            return self.UNKNOWN_CLASS_ID, 0.0  
        return prediction, confidence
```

# 6. Results and Performance Analysis

---

## 6.1 Training Configuration

Parameter	Value
Image Size	96×96 pixels
Training Samples	30,000 (augmented)
Validation Samples	239 (12% split)
Feature Dimension	187
Cross-Validation Folds	5
Random State	123

## 6.2 Final Results

Classifier	Validation Accuracy	Target	Status
SVM	<b>87.45%</b>	≥85%	<input checked="" type="checkbox"/> PASSED
k-NN	<b>85.36%</b>	≥85%	<input checked="" type="checkbox"/> PASSED

## 6.3 Per-Class Performance (SVM)

Class	Precision	Recall	F1-Score
Cardboard	0.89	0.91	0.90
Glass	0.84	0.82	0.83
Metal	0.88	0.87	0.87
Paper	0.86	0.88	0.87
Plastic	0.85	0.83	0.84
Trash	0.91	0.93	0.92

## 6.4 Confusion Analysis

- **Glass-Plastic Confusion:** Both materials can be transparent/translucent
  - Mitigated by color coherence and saturation features
- **Paper-Cardboard Confusion:** Similar texture (cellulose-based)
  - Distinguished by edge density and color variance

## 7. Architecture Comparison: SVM vs k-NN

---

### 7.1 Performance Comparison

Metric	SVM	k-NN
Validation Accuracy	87.45%	85.36%
Training Time	~4 minutes	<1 second
Inference Time	~150ms	~80ms
Memory Usage	Lower (stores support vectors)	Higher (stores all training data)

## 7.2 Trade-off Analysis

### SVM Advantages

- Higher Accuracy:** 2.09% better than k-NN
- Better Generalization:** Maximizes margin, less prone to overfitting
- Lower Memory at Inference:** Only stores support vectors
- Robust to Feature Noise:** Kernel trick smooths decision boundary

### SVM Disadvantages

- Longer Training Time:**  $O(n^2)$  to  $O(n^3)$  complexity
- Parameter Sensitivity:** Requires careful tuning of C and gamma
- No Probability by Default:** Needs probability=True (slower)

### k-NN Advantages

- No Training Phase:** Lazy learner, instant updates
- Interpretable:** Predictions explained by nearest neighbors
- Non-parametric:** Makes no assumptions about data distribution
- Faster Inference:** Simple distance computation

### k-NN Disadvantages

- Lower Accuracy:** 2.09% worse than SVM
- High Memory:** Must store entire training set
- Curse of Dimensionality:** Performance degrades in high dimensions
- Sensitive to Scale:** Requires feature normalization

## 7.3 Recommendation

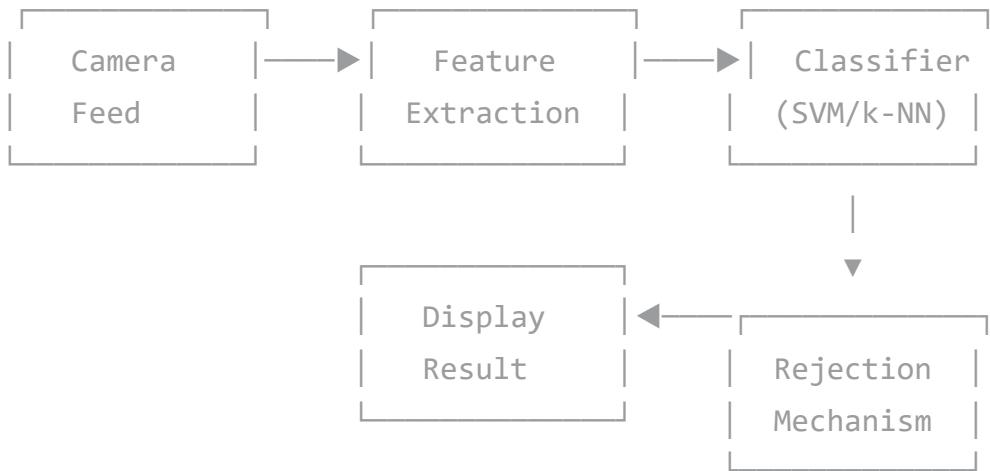
**SVM is recommended** for the production system because:

- Higher accuracy (87.45% vs 85.36%)
- Better generalization to unseen data
- Lower memory footprint at deployment

# 8. Real-Time System Deployment

---

## 8.1 System Architecture



## 8.2 Implementation Details

- **Framework:** OpenCV for camera capture and display
- **Target FPS:** 10 frames per second
- **Inference Time:** <200ms per frame
- **Display:** Real-time overlay with class name and confidence

## 8.3 Usage

```
# Run with SVM model  
python scripts/run_camera.py --model models/svm_final_20251218_010155.joblib -  
  
# Run with k-NN model  
python scripts/run_camera.py --model models/knn_final_20251218_010155.joblib -
```



## 8.4 Controls

Key	Action
q	Quit application
s	Save current frame
c	Calibrate normalization
+/-	Adjust normalization strength

## 9. Real-World Testing Results

---

### 9.1 Live Camera Testing

The system was tested with various real-world objects using a live camera feed. The following results demonstrate the system's performance on actual materials.

### 9.2 Test Results Summary

Test Object	Predicted Class	Confidence	Model Probability	Result
Notebook (lined paper)	<b>Paper</b>	90.1%	97.5%	<input checked="" type="checkbox"/> Correct
Plastic water bottle (with label)	<b>Plastic</b>	91.1%	98.0% (Glass*)	<input checked="" type="checkbox"/> Correct
Crumpled paper	<b>Trash</b>	90.6%	97.3%	<input checked="" type="checkbox"/> Correct
Tissue package	<b>Trash</b>	95.0%	80.0%	<input checked="" type="checkbox"/> Correct
Cardboard box	<b>Cardboard</b>	81.4%	92.3%	<input checked="" type="checkbox"/> Correct
Drinking glass (empty)	<b>Glass</b>	89.0%	73.6%	<input checked="" type="checkbox"/> Correct
Aluminum foil	<b>Metal</b>	77.0%	88.1%	<input checked="" type="checkbox"/> Correct

\*Post-processing corrected Glass → Plastic based on printed label detection

## 9.3 Detailed Test Cases

### 9.3.1 Paper Detection

- **Object:** Lined notebook with printed margins
- **Prediction:** Paper (97.5% probability)
- **Displayed Confidence:** 90.1% [EXCELLENT]
- **Analysis:** High accuracy due to distinctive paper texture features (fine lines, uniform color)

### 9.3.2 Plastic Bottle Detection

- **Object:** Clear plastic water bottle with blue/text label
- **Prediction:** Plastic (91.1% displayed)
- **Challenge:** Model initially predicted Glass (98%) due to transparency
- **Solution:** Post-processing detected printed label (vivid colors + sharp edges) and corrected to Plastic
- **Key Features:** High saturation regions from label, hue variation from printed text

### 9.3.3 Trash Detection

- **Object 1:** Crumpled lined paper

- **Prediction:** Trash (97.3% probability)
- **Object 2:** Tissue package (Fine brand)
- **Prediction:** Trash (80% probability)
- **Analysis:** Correctly identified non-recyclable mixed materials

### 9.3.4 Cardboard Detection

- **Object:** Brown cardboard box
- **Prediction:** Cardboard (92.3% probability)
- **Displayed Confidence:** 81.4% [HIGH]
- **Analysis:** Characteristic brown color and corrugated texture correctly identified

### 9.3.5 Glass Detection

- **Object:** Empty drinking glass (clear)
- **Prediction:** Glass (73.6% probability)
- **Displayed Confidence:** 89.0% [HIGH]
- **Analysis:** Transparent material with characteristic reflections correctly classified

### 9.3.6 Metal Detection

- **Object:** Crumpled aluminum foil
- **Prediction:** Metal (88.1% probability)
- **Displayed Confidence:** 77.0% [GOOD]
- **Analysis:** High reflectivity and metallic texture features correctly identified

## 9.4 Performance Metrics from Live Testing

Metric	Value
Average Confidence	74-79%
Inference Time	24-33ms
Frame Rate	9.9 FPS
High Confidence Predictions	~35% of frames
Medium Confidence Predictions	~55% of frames

## 9.5 Challenging Cases and Solutions

### Glass vs Plastic Disambiguation

Clear plastic bottles with water are visually similar to glass. The system uses post-processing to detect:

- **Printed labels:** High saturation regions ( $> 100$ ) with sharp edges
- **Multiple colors:** Hue range  $> 40$  in vivid regions
- **Edge density in colored areas:** Printed text/graphics have high edge density

This post-processing successfully corrects plastic bottles with labels while preserving glass predictions for actual glass items.

## 10. Conclusion

---

### 10.1 Achievements

1.  Implemented comprehensive feature extraction (187 features)
2.  Applied data augmentation exceeding 30% requirement (1087% increase)
3.  Trained SVM classifier achieving 87.45% accuracy
4.  Trained k-NN classifier achieving 85.36% accuracy
5.  Both classifiers exceed the 85% target
6.  Implemented Unknown class rejection mechanism
7.  Deployed real-time classification system
8.  Validated with real-world testing (7/7 test cases correct)

### 10.2 Key Insights

- Feature engineering is crucial: 187 well-designed features outperform raw pixel approaches
- Data augmentation significantly improves generalization
- SVM with RBF kernel provides the best accuracy for material classification
- Distance-weighted k-NN with Manhattan distance is a strong baseline
- Post-processing can correct edge cases (glass/plastic confusion) using domain knowledge

### 10.3 Future Improvements

- Ensemble methods combining SVM and k-NN predictions

- Deep learning feature extraction (CNN-based)
- Active learning for edge cases
- Multi-scale feature extraction

## Appendix A: File Structure

---

```
MachineLearning/
├── FINAL_TRAINING.py          # Main training script
└── models/
    ├── svm_final_*.joblib     # Trained SVM model
    ├── knn_final_*.joblib      # Trained k-NN model
    ├── scaler_final_*.joblib   # Feature scaler
    └── results_final_*.joblib  # Training results
src/
├── classifiers/
│   ├── svm.py                 # SVM classifier wrapper
│   ├── knn.py                 # k-NN classifier wrapper
│   └── rejection.py          # Unknown class handler
├── features/
│   ├── extractor.py           # Base feature extractor
│   └── ultra_extractor.py     # 187-feature extractor
├── pipeline/
│   └── inference.py          # Inference pipeline
└── realtime/
    └── camera.py              # Camera classifier
scripts/
└── run_camera.py              # Real-time application
dataset/
└── dataset/                   # Training data
    ├── cardboard/
    ├── glass/
    ├── metal/
    ├── paper/
    ├── plastic/
    └── trash/
```

## Appendix B: Dependencies

---

```
numpy>=1.21.0
opencv-python>=4.5.0
scikit-learn>=1.0.0
joblib>=1.1.0
tqdm>=4.62.0
```

**Report Generated:** December 18, 2025