Cairo University
Faculty of Computers and Artificial Intelligence

Machine Learning Project Report

Material Stream Identification System

By

Name	ID
Nouran Ashraf Ali	20221242
Jana Tarek Mohamed	20221215
Maya Mohamed	20220271
Abdallah Mohamed Ali	20220185
Roqaia Hassan	20220129

Dr. Hanaa Bayomi

TA. Mohamed Ata

1. Data Augmentation

The original dataset was unbalanced some material classes had way more images than others. That's a problem for any classifier, since it tends to favor the bigger classes and struggles with rare ones. To fix this, we used data augmentation to bulk up the smaller classes. The goal was to get each class up to 500 images.

Leaned on ImageDataGenerator and set it up with a few tricks:

- random rotations (plus or minus 15 degrees)
- horizontal and vertical shifts (up to 10%)
- zooming in and out a bit (again, around 10%), and horizontal flips.

For any empty pixels that popped up, we just filled them in with the nearest neighbors. These moves make the images look as if they were taken from different angles, at different scales, or with a shaky hand to be pretty close to what you'd see in real life.

Augmenting was kept until each class hit 500, to not end up with a bunch of near-duplicates that could lead to overfitting.

By the end, the dataset grew by over 30%. This met the required quota and made sure every class got equal representation.

2. Feature Extraction (Image to Vector Conversion)

To turn raw images into something a machine learning model can use, that's needed to extract features (translate pictures into numbers).

For that, MobileNetV2 was chosen, a neural network that comes pre-trained and is known for being light and fast.

- Steps done:
1. First, resize all images to 128 by 128 pixels, with 3 color channels.
2. Then, preprocessed them using MobileNetV2's own method.
3. Next, feed the images through MobileNetV2 but left off the final classification layer.
4. Finally, we used global average pooling to squash everything down to a neat, one-dimensional vector.

- Why MobileNetV2?

Prior to using MobileNetV2, we experimented with HOG as a handcrafted feature descriptor.

Findings:

- Using HOG features with SVM resulted in a validation accuracy of 60%, significantly lower than the 94% achieved with MobileNetV2.
- HOG struggled to capture complex texture and color patterns present in real-world material images.
- HOG features were sensitive to lighting variations, rotations, and scale changes, reducing generalization.

SO MobileNetV2 is fast, doesn't eat up much memory, and still grabs the important stuff as textures, edges, patterns that matter for telling materials apart.

Plus, with global average pooling, we got a compact vector for each image, which plays nicely with classic machine learning models. It's how you bridge the gap between deep learning and more traditional machine learning

3. Model Architecture and Implementation

By trained two core machine learning models on those feature vectors: Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN).

3.1 Support Vector Machine (SVM)

The setup:
- Kernel: Radial Basis Function (RBF)
- C: 10
- Gamma: scale
- Probability estimation turned on

Why SVM?

- The RBF kernel handles non-linear decision boundaries effectively.
- SVMs perform well in high-dimensional feature spaces, which is ideal for CNN-based features.
- Margin maximization improves generalization.
- Probability outputs allow confidence-based rejection for Unknown class handling.

Results:
- Validation Accuracy: 94.14%
- Macro F1-score: 0.92
- Unknown Class Recall: 0.99

SVM stayed consistent across all materials and handled tricky, ambiguous cases really well.

3.2 k-Nearest Neighbors (k-NN)

Setup:
- k = 5
- Distance-weighted voting
- StandardScaler for feature normalization

Why k-NN?

- k-NN is simple, interpretable, and effective for multi-class classification.
- Distance weighting reduces the influence of distant neighbors.
- Feature scaling is essential to ensure fair distance computation.

Results:
- Validation Accuracy: 87.29%
- Macro F1-score: 0.82
- Unknown Class Precision: 0.52

k-NN did alright with clear-cut classes, but it struggled more when things got noisy or overlapped.

Comparison:

Comparing the results of the both method:

```
    Metric                    SVM             k-NN
    ------------------------------------------------------------
••• Accuracy                  0.92 0.8142857142857143
    Validation accuracy: 0.9414285714285714 0.8728571428571429
```

For SVM:

```
  classification_report:
                precision    recall  f1-score   support

       Unknown       0.79      0.99      0.88       100
     cardboard       0.96      0.98      0.97       100
         glass       0.97      0.90      0.93       100
         metal       0.93      0.87      0.90       100
         paper       0.92      0.87      0.89       100
       plastic       0.95      0.88      0.91       100
         trash       0.98      0.95      0.96       100

      accuracy                           0.92       700
     macro avg       0.93      0.92      0.92       700
  weighted avg       0.93      0.92      0.92       700
```

For KNN:

```
classification_report:
              precision    recall  f1-score   support

     Unknown       0.52      0.89      0.65       100
   cardboard       0.94      0.90      0.92       100
       glass       0.88      0.79      0.83       100
       metal       0.94      0.67      0.78       100
       paper       0.89      0.77      0.82       100
     plastic       0.93      0.74      0.82       100
       trash       0.90      0.94      0.92       100

    accuracy                           0.81       700
   macro avg       0.86      0.81      0.82       700
weighted avg       0.86      0.81      0.82       700
```

Insights:

SVM achieved superior accuracy (94.1% validation) compared to k-NN (87.3%).

Macro F1-score indicates that SVM maintains balanced performance across all classes, whereas k-NN shows reduced effectiveness, particularly on classes with overlapping visual features.

SVM consistently performed well across all primary classes (precision and recall ≥ 0.87), with minor variability in classes such as glass and metal.

k-NN showed lower recall for classes like metal (0.67) and plastic (0.74), suggesting sensitivity to feature similarity and noisy boundaries in high-dimensional space.

Conclusion:

The comparative analysis demonstrates that SVM is the superior classifier for this material identification system. It achieves higher accuracy, balanced performance across classes, and robust Unknown detection, making it the preferred choice for real-time deployment.

4. Handling the “Unknown” Class (ID 6)

Used a two-step approach:

4.1 Synthetic Unknown Generation

Created artificial Unknowns by mixing in random noise images and heavily blurring some real images. The idea was to mimic stuff the model hadn’t seen before

4.2 Confidence-Based Rejection Mechanism

During prediction, set up a rule: if the model’s top-class probability was below 0.6, it labeled the image as Unknown. Only confident predictions were assigned to a material class. This helped prevent the model from forcing a bad guess when it wasn’t sure.

SVM stood out here, hitting 99% recall for the Unknown class and still keeping its accuracy high on known materials.

5. System Deployment (Real-Time Classification)

We took our top model:

MobileNetV2 paired with SVM, and plugged it straight into a real-time camera app. Here’s how it works: the camera grabs live frames, resizes and preprocesses them, then runs them through the CNN to pull out features. After that, the SVM classifier kicks in, checks confidence, and spits out both the predicted label and its confidence score right on the screen.