TraceFinder - Forensic Scanner Identification

Project Statement:

The aim of this project is to identify the source scanner device used to scan a document or image by analyzing the unique patterns or artifacts left behind during the scanning process. Each scanner (brand/model) introduces specific noise, texture, or compression traces that can be learned by a machine learning model. This project is important in forensic investigations, copyright authentication, and document verification tasks.

Use Cases:

Digital Forensics

- Description: Determine which scanner was used to forge or duplicate legal documents.
- Example: Detect whether a fake certificate was created using a specific scanner model.

Document Authentication

- Description: Identify the source of printed and scanned images to detect tampering or fraudulent claims.
- Example: Differentiate between scans from authorized and unauthorized departments.

Legal Evidence Verification

- Description: Ensure scanned copies submitted in court/legal matters came from known and approved devices.
- Example: Verify that scanned agreements originated from the company's official scanner.

Outcomes:

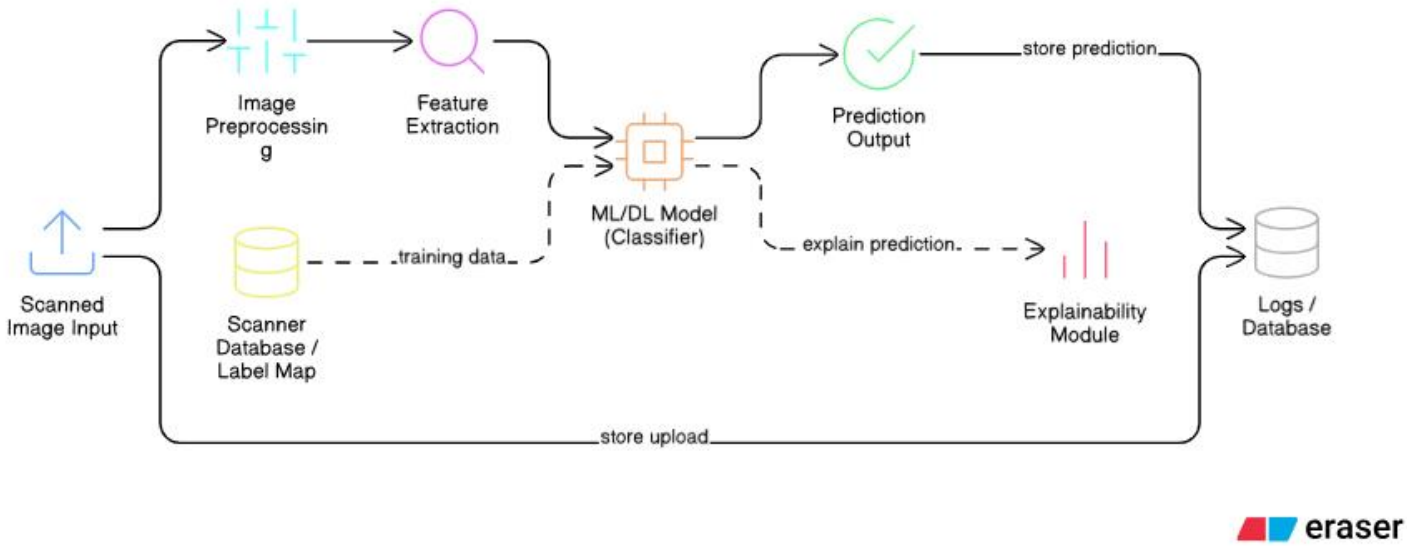
By the end of this project, students will:

- Understand the concept of source device identification.
- Extract scanner-specific features such as noise patterns, frequency domain signals, and artifacts.
- Train a classification model to distinguish among multiple scanners.
- Evaluate model accuracy and visualize feature importance.
- Optionally deploy a simple app to upload and identify the scanner source.

Dataset:

Source: Kaggle

System Architecture



Modules to Be Implemented:

1. Data Collection & Labeling

- Manually scan sample images using multiple scanners
- Assign proper labels based on source device

2. Image Preprocessing

- Resize, denoise, convert to grayscale if needed
- Normalize pixels and remove non-artifact content

3. Feature Extraction

- Extract noise patterns using filters (wavelet, FFT)
- Compute PRNU, texture descriptors, edge patterns

4. Model Training

- Train classifiers like:
    - CNN (if using deep features)
    - Random Forest, SVM (if using extracted features)
- Evaluate on validation set

5. Output System

- Upload an image → return probable scanner model
- Optional: Show confidence score + feature map

Week-wise module implementation and high-level requirements with output screenshots

Milestone 1: Dataset Collection & Preprocessing

Week 1:

- Collect scanned document samples from different scanner devices (minimum 3–5 models/brands).
- Create a labeled dataset (e.g., scanner_model, file_name, etc.).
- Analyze basic image properties: resolution, format, color channel, etc.

Week 2:

- Perform image preprocessing:
    - Resize all images to a fixed dimension
    - Convert to grayscale if needed
    - Denoise (optional)
- Normalize and structure the dataset for model training.

Milestone 2: Feature Engineering & Baseline Modeling

Week 3:

- Extract hand-crafted features like:
    - Noise patterns
    - Frequency domain features (e.g., FFT)
    - Texture descriptors (e.g., LBP)
- Visualize differences between scanner outputs (e.g., noise maps).

Week 4:

- Train baseline ML models (e.g., Logistic Regression, SVM, Random Forest).
- Evaluate using accuracy and confusion matrix.
- Log performance and identify limitations of hand-crafted features.

Milestone 3: Deep Learning Model + Explainability

Week 5:

- Build and train a CNN model on the raw image dataset.
- Use image augmentation for generalization (brightness, rotation).
- Tune hyperparameters and track training curves (accuracy/loss).

Week 6:

- Evaluate model performance: accuracy, F1-score, confusion matrix.
- Apply explainability tools (e.g., SHAP or Grad-CAM) to visualize how the model identifies scanner-specific patterns.

Milestone 4: Deployment & Final Report

Week 7:

- Create a simple **UI using Streamlit** (or any frontend):
    - Upload scanned image
    - Get predicted scanner brand/model with confidence score
- Log predictions and allow download of result.

Week 8:

- Final documentation and formatting
- Add system architecture, training results, model comparison, and screenshots.
- Prepare final presentation slides and demonstrate the working model.

Evaluation Criteria

Completion of Tasks:

- Data collected and labeled correctly
- Feature engineering and modeling completed
- UI integration and testing done

Model Quality:

- Classification accuracy above baseline (ideally >85%)
- Model should distinguish between at least 3–5 scanners
- Robustness to image format and scan resolution

Documentation & Demo:

- Clear explanation of methodology
- Model performance charts (accuracy, confusion matrix)
- Insight into what features helped most (explainability)

Model Performance – Quantitative Metrics

Classification Metrics:

- **Accuracy**: Correct scanner prediction rate
- **Precision**: Correct predictions for each scanner class
- **Recall**: Ability to detect all true scanner outputs
- **F1-Score**: Balance of precision and recall
- **Confusion Matrix**: Visual of misclassification between scanner types

Feature Analysis:

- Use SHAP or Grad-CAM (if CNN used) to show key areas
- Validate model isn't biased toward scan brightness or layout