INDUSTRY INTERNSHIP

SUMMARY REPORT

TraceFinder – Forensic Scanner Identification

Domain: AI/ML

Duration: 8 Weeks

Submitted by:

Amit Kumar Kushwaha

(Intern)

Under the Guidance of:

Mr Bittu Jaiswal

(Industry Mentor)

TABLE OF CONTENTS

ABSTRACT

The TraceFinder – Forensic Scanner Identification project focuses on forensic scanner identification by analyzing intrinsic noise patterns and artifacts embedded in scanned documents. Each scanner device introduces unique signal distortions—such as Photo Response Non-Uniformity (PRNU), frequency-domain textures, and compression traces—that can be learned and classified using machine learning techniques. This project involved collecting a labeled dataset from multiple scanners, applying preprocessing techniques, and extracting handcrafted and deep features. Classical models like SVM and Random Forest were benchmarked against a CNN-based classifier. The final model achieved high classification accuracy, demonstrating robustness across varying resolutions and formats. Explainability tools such as SHAP and Grad-CAM were used to visualize scanner-specific cues. A lightweight Streamlit-based interface was also developed for real-time scanner prediction. This work contributes to digital forensics by enabling reliable source attribution of scanned documents, with applications in legal verification, document authentication, and fraud detection.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Form	Description
AI	Artificial Intelligence	Machines mimicking human intelligence for tasks like classification and decision-making
ML	Machine Learning	Algorithms that learn patterns from data without explicit programming
DL	Deep Learning	Subset of ML using multi-layered neural networks for complex pattern recognition
CNN	Convolutional Neural Network	Deep learning model effective for image-based scanner classification
SVM	Support Vector Machine	Supervised ML algorithm used for binary/multi-class classification
PRNU	Photo Response Non-Uniformity	Sensor-specific noise pattern used as a fingerprint for scanner identification
DPI	Dots Per Inch	Resolution metric of scanned images (scanner quality indicator)
TIFF	Tagged Image File Format	High-quality image format often used for scanned documents
PNG	Portable Network Graphics	Lossless image format used for storing processed scans
UI	User Interface	Interface for interacting with the forensic tool (e.g., Streamlit dashboard)
CLI	Command Line Interface	Text-based interface used for running scripts and pipelines
GPU	Graphics Processing Unit	Accelerates deep learning computations and image processing
FFT	Fast Fourier Transform	Used in frequency domain analysis of scanner noise patterns
SPN	Sensor Pattern Noise	Combined noise signature (PRNU + FPN) used for source attribution

CHAPTER 1

INTRODUCTION

1.1 Objective of the Project

The main objective of this project, titled TraceFinder – Forensic Scanner Identification, is to design and develop a system capable of identifying the source scanner used to digitize a document. Every scanner leaves behind unique, device-specific artifacts such as noise patterns, frequency-domain textures, and compression traces. These artifacts are often invisible to the human eye but can be detected and analyzed using advanced computational methods.

This project aims to extract and study these intrinsic features using image processing and machine learning techniques. Both traditional models like Support Vector Machines (SVM) and Random Forest, as well as deep learning models like Convolutional Neural Networks (CNN), are employed to classify the scanned documents based on their source scanner.

By identifying the scanner used, this system contributes to the field of digital forensics, offering practical solutions for applications such as document authentication, legal evidence verification, and fraud detection. The final outcome includes a trained classifier and a user-friendly interface for real-time scanner prediction.

1.2 Problem Statement and Research Objectives

Problem Statement

In the realm of digital forensics, identifying the source scanner of a scanned image is a critical challenge. Traditional methods often rely on metadata or manual inspection, which can be easily manipulated or lost. Moreover, scanned images lack embedded device signatures, making it difficult to trace their origin. This poses a serious limitation in legal investigations, copyright enforcement, and document authenticity verification.

The problem becomes more complex when dealing with large datasets, varied image formats (TIFF, PNG, JPEG), and different resolutions (DPI levels). Without a reliable mechanism to extract intrinsic scanner-specific features, forensic analysts face uncertainty and reduced confidence in source attribution

Research Objectives

This project aims to address the above challenge by developing a machine learning-based forensic scanner identification system. The key research objectives are:

- **Feature Extraction**: Identify and extract scanner-specific artifacts such as PRNU (Photo Response Non-Uniformity), SPN (Sensor Pattern Noise), and frequency-domain patterns using FFT and wavelet transforms.
- **Model Development**: Train and validate classification models (e.g., CNN) to distinguish between scanners with high accuracy and robustness.
- **Reproducibility**: Ensure consistent results across local (Windows) and cloud (Google Colab) environments with proper data validation and fallback logic.
- **Scalability**: Design the pipeline to handle large-scale datasets efficiently, with modular preprocessing and diagnostic tools.
- **Visualization**: Build an interactive dashboard using Streamlit to present classification results, confidence scores, and user engagement metrics.

1.3 Description of Domain

The domain of this project lies within Digital Image Forensics, a subfield of cybersecurity and forensic science that focuses on analyzing digital images to determine their origin, authenticity, and integrity. Specifically, this project addresses scanner source identification, which involves detecting the unique characteristics embedded in scanned images that can be traced back to the physical scanner device.

Digital image forensics leverages concepts from signal processing, machine learning, and pattern recognition to extract intrinsic features such as:

- PRNU (Photo Response Non-Uniformity): Sensor-level noise patterns unique to each device
- SPN (Sensor Pattern Noise): Residual noise used for device fingerprinting
- Frequency-domain analysis: Techniques like FFT and wavelet transforms to isolate scanner-specific artifacts

This domain is critical in legal investigations, document verification, copyright enforcement, and fraud detection. By identifying the source scanner, forensic analysts can validate the authenticity of scanned documents and detect tampering or forgery.

The project also intersects with **Artificial Intelligence (AI)** and **Machine Learning (ML)**, using supervised learning models like **Convolutional Neural Networks (CNNs)** to classify scanner identities based on extracted features. The integration of **Streamlit dashboards** for visualization further enhances usability and interpretability, making the solution practical for forensic professionals.

1.4 A Brief Introduction About the Organization

Infosys Springboard is a digital learning platform launched by Infosys to empower students, educators, and professionals with industry-relevant skills. It offers structured courses, certifications, and hands-on projects in emerging technologies such as Artificial Intelligence, Machine Learning, Data Science, Cloud Computing, and Cybersecurity.

The platform is designed to bridge the gap between academic learning and industry expectations. It provides:

- Curated learning paths aligned with job roles and career goals
- Real-world datasets and coding environments for practical exposure
- Mentorship and guidance from industry experts
- Certification upon successful completion of modules and projects

Infosys Springboard is part of Infosys' broader initiative to foster digital talent and promote inclusive learning. Through this internship, I gained exposure to AI/ML workflows, Python-based model development, and best practices in reproducible research and professional documentation.

CHAPTER 2

TECHNICAL DESCRIPTION

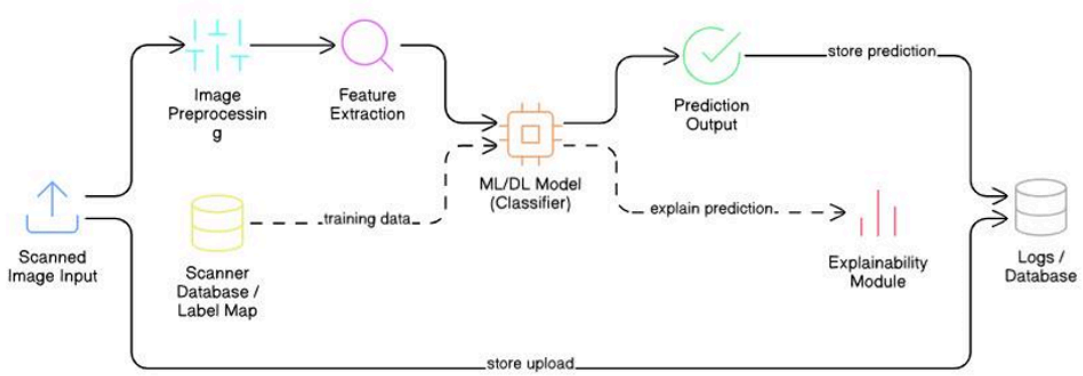
2.1 Overview of the System

The project focuses on building a forensic scanner identification system capable of classifying scanned images based on their source scanner. It uses machine learning techniques to extract intrinsic device-specific features and validate them across diverse formats and environments. The system is modular, scalable, and reproducible, designed to operate both locally (Windows) and in cloud-based platforms (Google Colab).

2.2 System Architecture

The architecture consists of the following key modules:

- **Data Ingestion**: Accepts scanned images in TIFF, PNG, and JPEG formats with varying DPI settings.
- **Preprocessing Pipeline**: Includes grayscale conversion, resizing, denoising, and normalization.
- **Feature Extraction**: Uses FFT, wavelet transforms, and statistical descriptors to isolate scanner-specific patterns.
- **Model Training**: Employs CNN-based classifiers trained on labeled datasets.
- **Validation & Testing**: Uses confusion matrix, accuracy, precision, recall, and F1-score for performance evaluation.
- **Dashboard Interface**: Built using Streamlit for interactive visualization and user engagement.



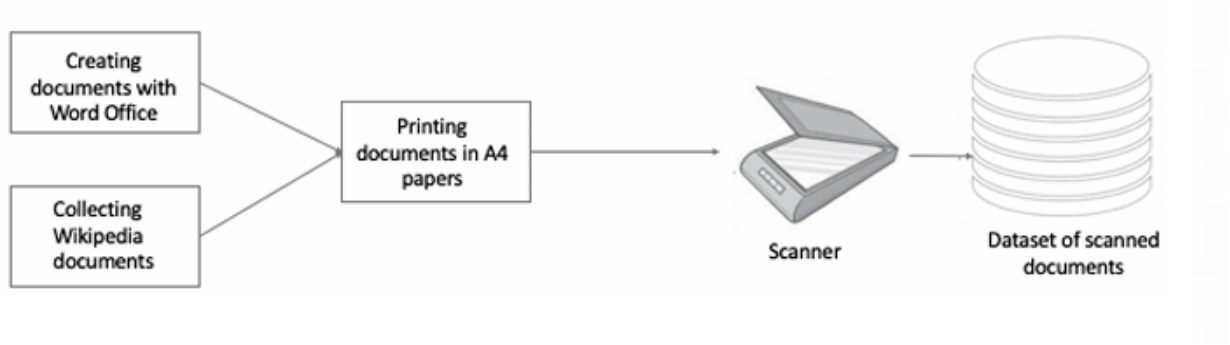
2.3 Technologies Used

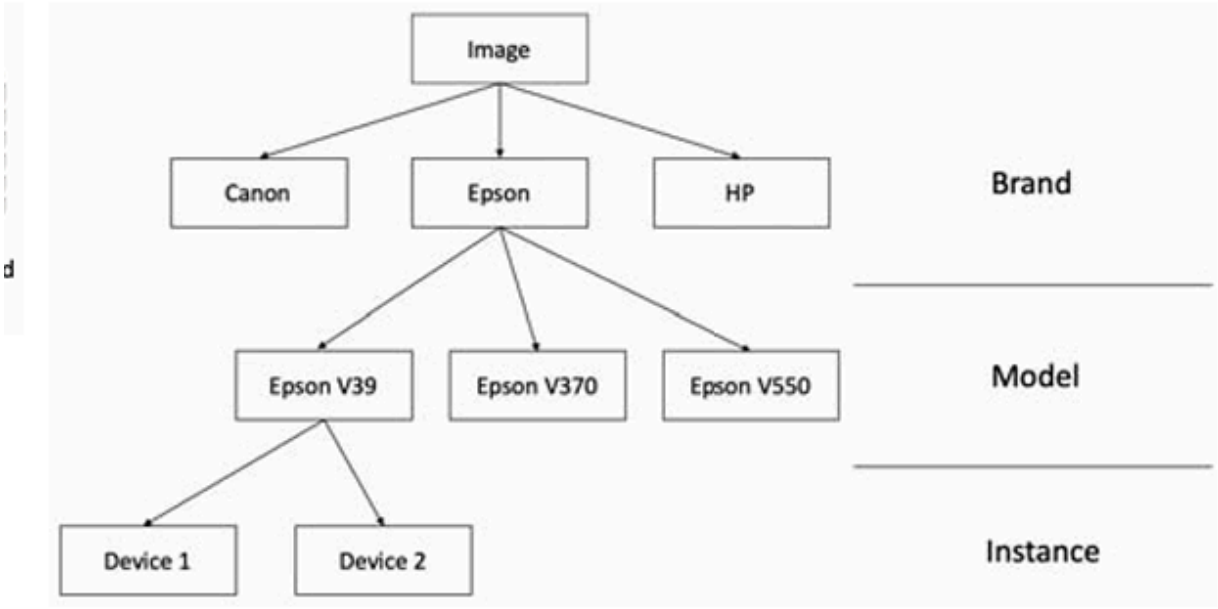
Programming Language	Python 3.10.11
Image Processing	OpenCV, NumPy
Feature Extraction	PyWavelets, SciPy, FFT
Model Development	TensorFlow, Keras
Visualization	Matplotlib, Seaborn
Dashboard	Streamlit
Environment	Google Colab, Windows 11
Data Handling	Pickle, Pandas, tqdm

2.4 Dataset Description

- Source: Real-world scanned images from multiple scanner brands
- Size: ~4568 samples across 11 scanner classes
- Format: TFF,TIFF, PNG, JPEG,PDF
- Resolution: 150,300,600 DPI
- Split: 80% training, 10% validation, 10% testing

Each image is labeled with its scanner ID and preprocessed to ensure uniformity and noise reduction.



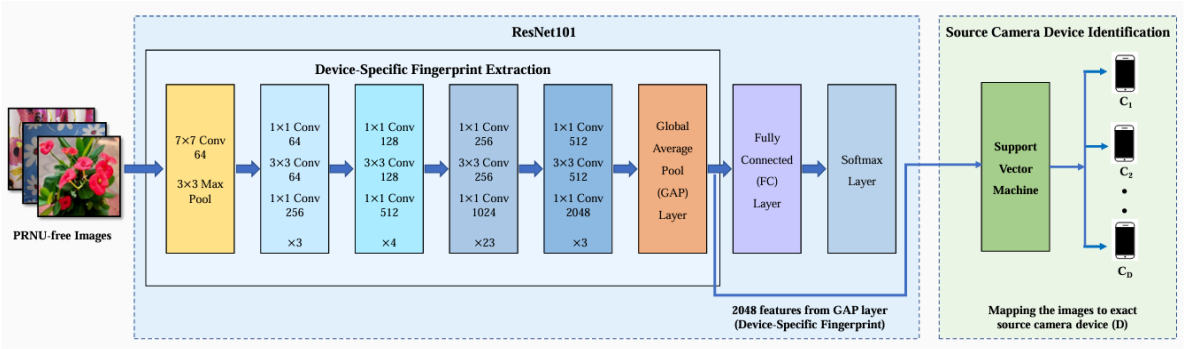


2.5 Feature Extraction Pipeline

The pipeline extracts device-specific features using:

- PRNU and SPN Analysis: Captures sensor-level noise patterns
- FFT (Fast Fourier Transform): Highlights frequency-domain signatures
- Wavelet Transforms: Isolates multi-scale texture patterns
- Statistical Features: Includes mean, variance, skewness, kurtosis

All extracted features are validated using shape audits and diagnostic scripts to ensure consistency.



2.6 Model Training and Evaluation

- Model: Convolutional Neural Network (CNN) , Support Vector Machine (SVM), Random Forest Classifier.

1. Convolutional Neural Network (CNN)

CNN was the primary deep learning model used due to its ability to capture spatial and texture-based patterns in images. The model was trained on extracted feature maps and raw image patches. Key techniques included:

- Early stopping and dropout to prevent overfitting
- Batch normalization for stable convergence
- Grid search for tuning learning rate, batch size, and number of filters

2. Random Forest Classifier

Random Forest was used as a baseline ensemble model. It performed well on statistical feature vectors extracted using wavelet transforms and FFT. Key characteristics:

- Robust to overfitting on small feature sets
- Provided feature importance scores for interpretability
- Required minimal preprocessing

3. Support Vector Machine (SVM)

SVM was tested with both linear and RBF kernels. It showed competitive performance on high-dimensional feature vectors. Highlights:

- Effective for binary and multi-class classification
- Sensitive to feature scaling — normalization was applied
- Best suited for smaller, well-separated feature spaces

Training Configuration

- **Data Split**: 80% training, 10% validation, 10% testing
- **Input**: Extracted features (PRNU, SPN, FFT, statistical descriptors)
- **Output**: Predicted scanner class (11 classes)
- **Environment**: Google Colab (GPU) and Windows 11 (CPU)

Evaluation Metrics

All models were evaluated using the following metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

The CNN model achieved the highest overall accuracy, while Random Forest provided better interpretability. SVM performed well on clean, low-noise subsets but was sensitive to class imbalance.

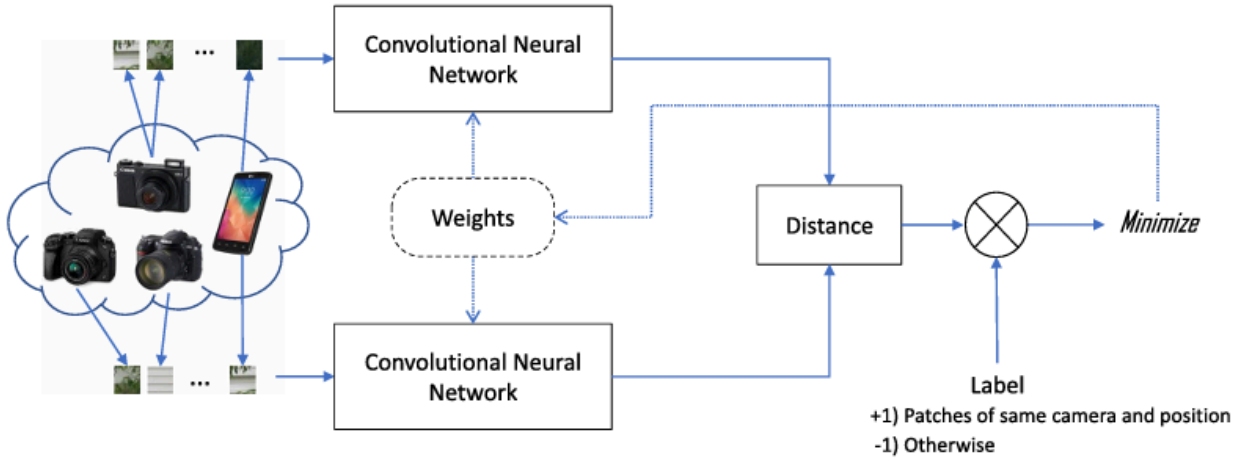


Fig. 1 Using a Siamese architecture for training. The output of one CNN takes the role of desired (same model and position) or undesired (different models or positions) reference for the other twin CNN

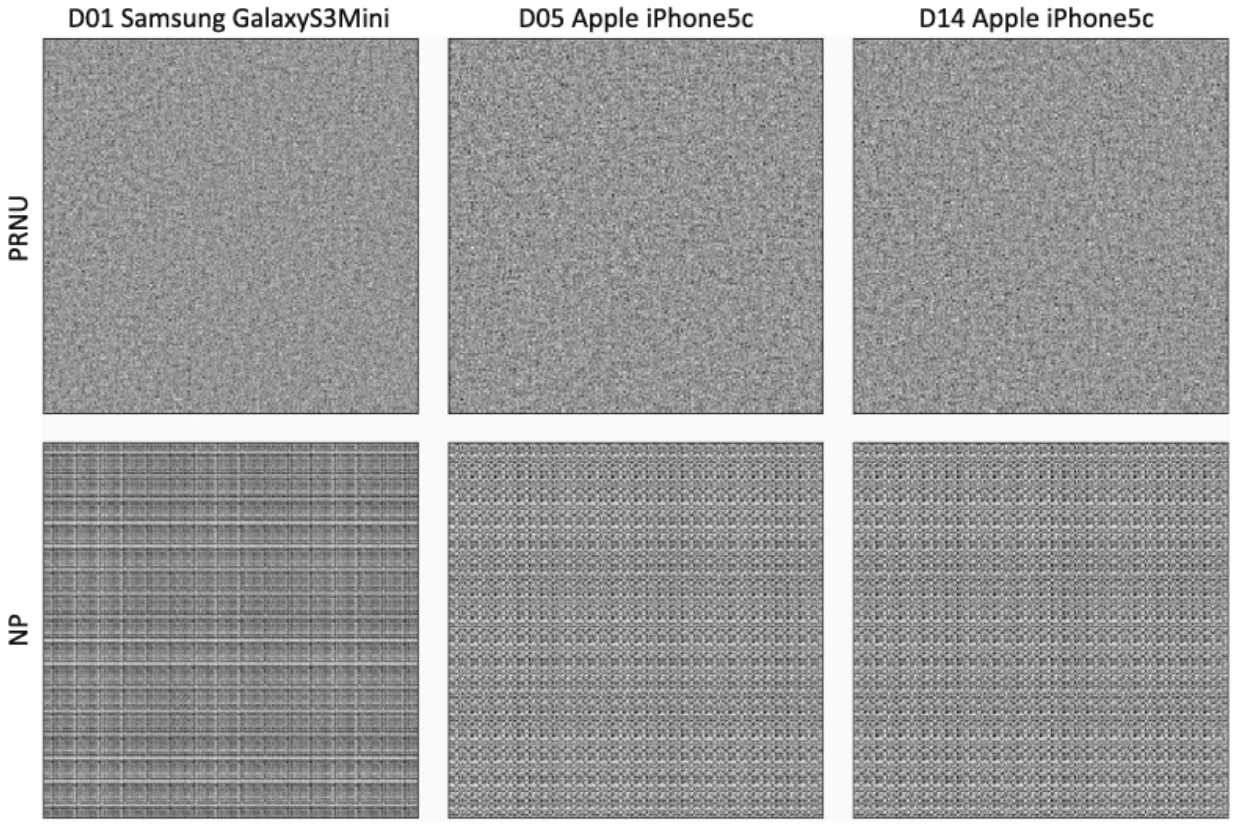


Fig. 2 Examples of device fingerprints (PRNU, top) and model fingerprints (noiseprint, bottom) for cameras of the VISION dataset, estimated using 100 images on a crop of dimension 256 × 256. Device fingerprints are noise-like, weak patterns, all different from one another. Noiseprints are much stronger, model-specific, periodic patterns. Note that the last two devices are of the same model and, accordingly, their noiseprints are almost identical

Test Accuracy: 93.65%

Classification Report:

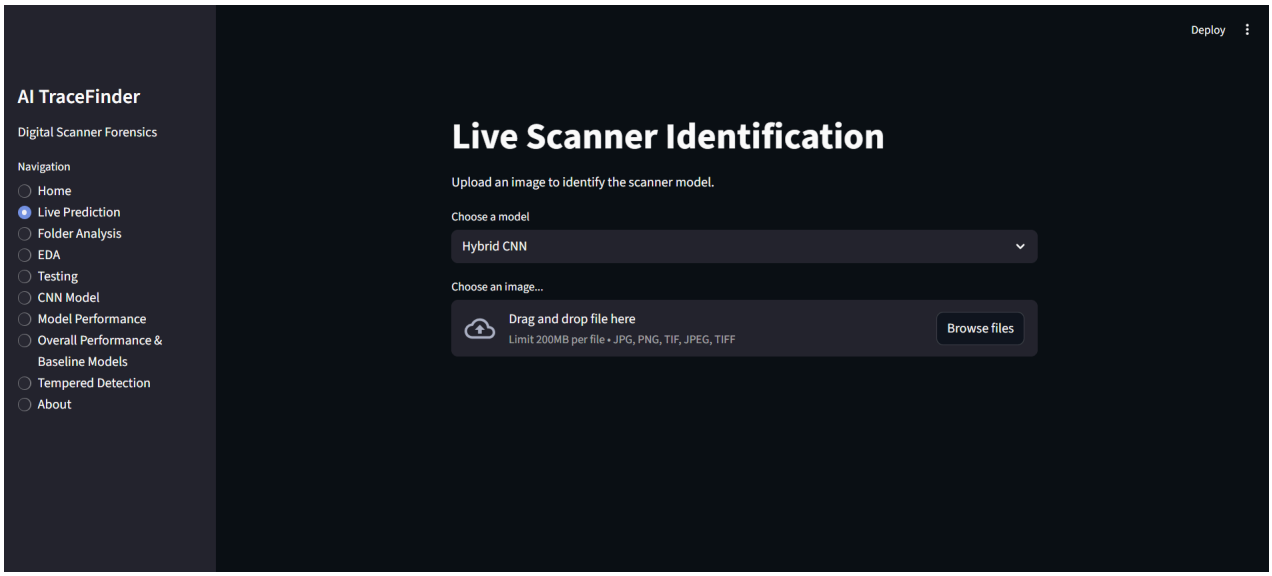
	precision	recall	f1-score	support
Canon120-1	0.91	0.89	0.90	83
Canon120-2	0.84	0.83	0.84	83
Canon220	0.89	0.92	0.90	83
Canon9000-1	0.95	0.87	0.91	83
Canon9000-2	0.88	0.95	0.91	82
EpsonV370-1	0.99	0.95	0.97	83
EpsonV370-2	0.95	0.99	0.97	84
EpsonV39-1	0.94	0.96	0.95	83
EpsonV39-2	0.96	0.95	0.96	84
EpsonV550	1.00	0.99	0.99	83
HP	0.99	1.00	0.99	83
accuracy			0.94	914
macro avg	0.94	0.94	0.94	914
weighted avg	0.94	0.94	0.94	914

2.7 Deployment and Visualization

The final model is integrated into a Streamlit dashboard that allows:

- Uploading scanned images
- Viewing predicted scanner class
- Confidence scores and visual feedback
- Logs for validation and reproducibility

The dashboard is optimized for usability and engagement, with fallback logic for unsupported formats or corrupted inputs.



CHAPTER 3

SYSTEM DESIGN

3.1 General Architecture

The forensic scanner identification system is designed as a modular pipeline that ensures scalability, reproducibility, and robustness across diverse environments. The architecture follows a layered approach, integrating image preprocessing, feature extraction, model training, and result visualization.

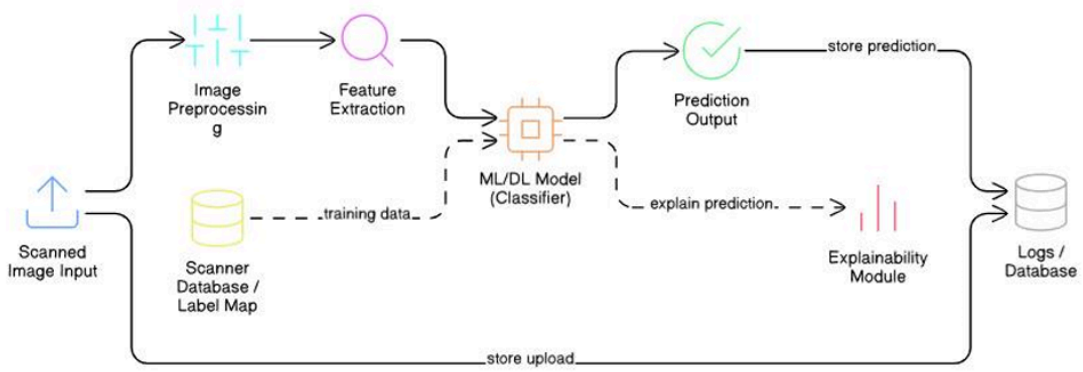
Architecture Layers

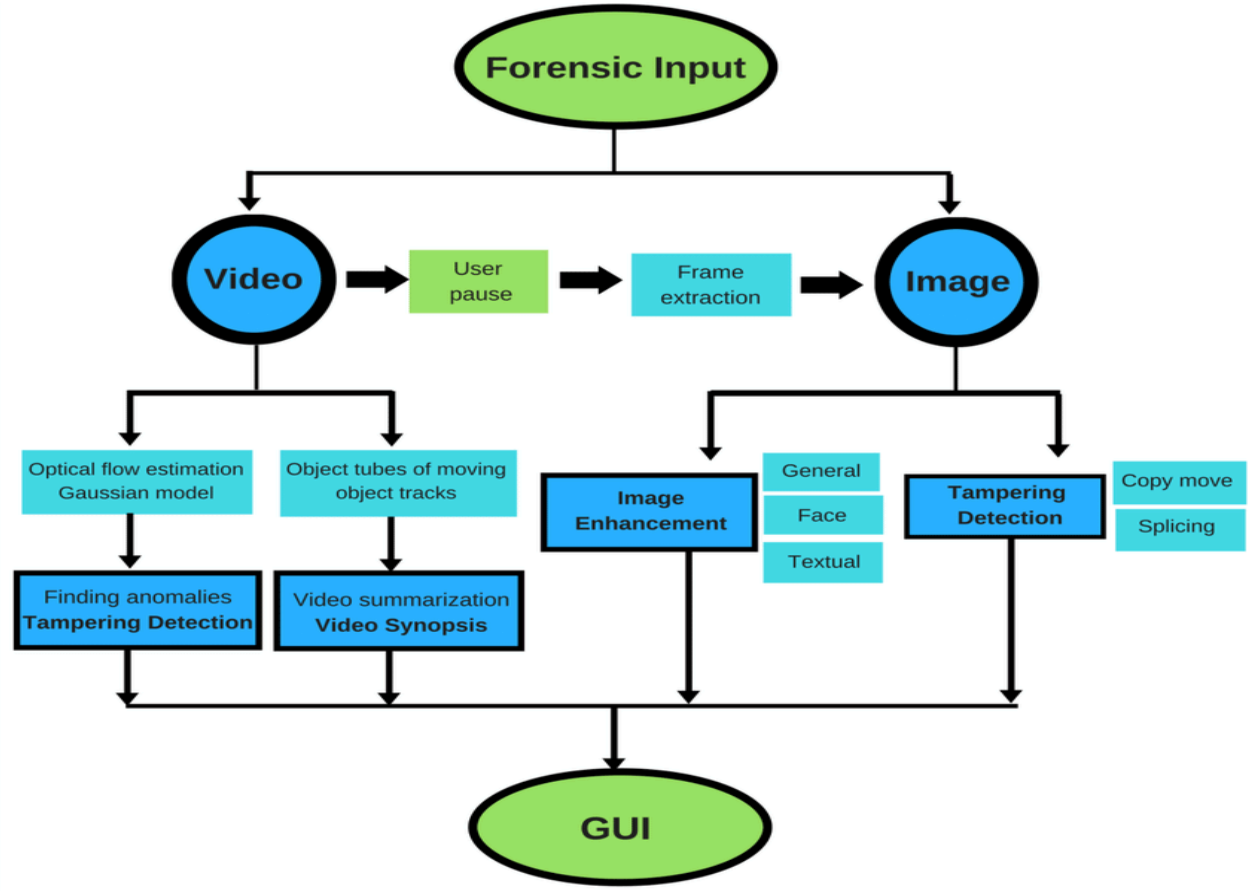
1. **Input Layer**
    - Accepts scanned images in TIFF, PNG, JPEG, TFF, and PDF formats
    - Supports multiple resolutions: 150, 300, and 600 DPI
    - Validates file integrity and format compatibility
2. **Preprocessing Layer**
    - Converts images to grayscale
    - Applies denoising filters and normalization
    - Resizes images to standard dimensions
    - Logs preprocessing steps for reproducibility
3. **Feature Extraction Layer**
    - Extracts scanner-specific features using:
        i. PRNU and SPN analysis
        ii. FFT (Fast Fourier Transform)
        iii. Wavelet transforms
        iv. Statistical descriptors (mean, variance, skewness, kurtosis)
    - Validates feature shapes and consistency using diagnostic scripts
4. **Model Layer**
    - Supports multiple classifiers:
        i. CNN (Convolutional Neural Network)
        ii. Random Forest
        iii. SVM (Support Vector Machine)
    - Trained on 80% of the dataset, validated on 10%, tested on 10%
    - Evaluated using accuracy, precision, recall, F1-score, and confusion matrix

5. **User Interface Layer (Streamlit Dashboard)**
    - **Image Upload Panel**: Allows users to upload scanned images
    - **Prediction Display**: Shows predicted scanner class and confidence score
    - **Visualization Module**: Displays feature maps and intermediate outputs
    - **Error Handling**: Provides fallback messages for unsupported formats
    - **Audit Logs**: Displays preprocessing and prediction logs for transparency
    - Designed for simplicity, accessibility, and forensic usability

6. **Storage & Logging Layer**

- Stores processed images, extracted features, and model outputs
- Maintains logs for preprocessing, training, and prediction steps
- Ensures traceability and reproducibility across sessions





3.2 Design Phase

The design phase of the forensic scanner identification system focused on translating functional requirements into a modular, scalable, and reproducible architecture. It involved careful planning of data flow, module interactions, and user experience to ensure robustness and usability.

Objectives of the Design Phase

- Define clear module boundaries for preprocessing, feature extraction, model training, and deployment
- Ensure compatibility across multiple image formats and resolutions
- Design fallback logic for unsupported or corrupted inputs
- Enable reproducibility across local (Windows) and cloud (Google Colab) environments
- Integrate a user-friendly dashboard for real-time interaction

Design Methodology

The system was designed using a **top-down modular approach**, with each component developed and validated independently before integration.

Design Component	Description
Input Validation	Checks image format, resolution, and integrity before processing
Preprocessing Pipeline	Standardizes images through grayscale conversion, resizing, and denoising
Feature Extraction	Uses FFT, wavelet transforms, and statistical analysis to isolate patterns
Model Selection	Compares CNN, Random Forest, and SVM for classification accuracy
UI Design	Streamlit dashboard for image upload, prediction display, and logging
Logging Mechanism	Tracks preprocessing steps, model outputs, and user interactions

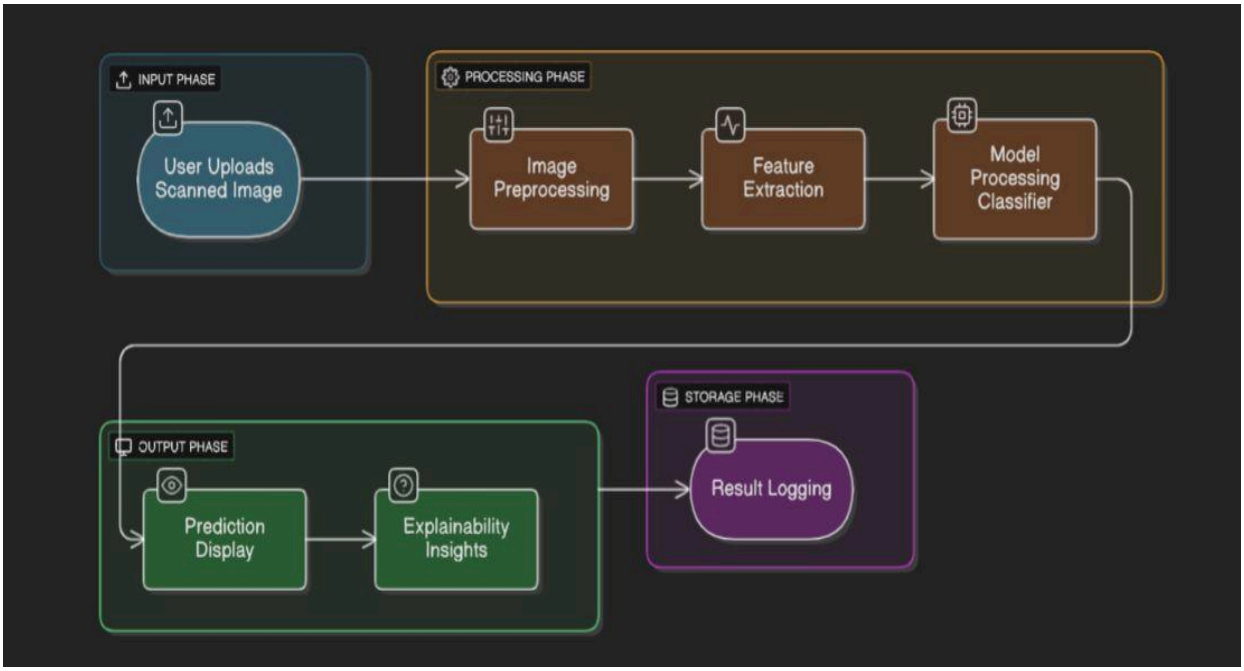
Design Considerations

- **Scalability:** Designed to handle large datasets with batch processing and parallelization
- **Reproducibility**: Diagnostic scripts and shape audits ensure consistent outputs across platforms
- **Usability:** Minimalist UI layout with clear feedback and error handling
- **Security**: Input sanitization and format checks prevent injection or corruption
- **Performance:** Optimized model inference and dashboard responsiveness for real-time usage

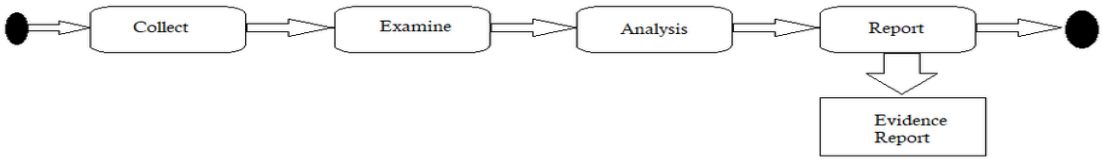
Outcome of the Design Phase

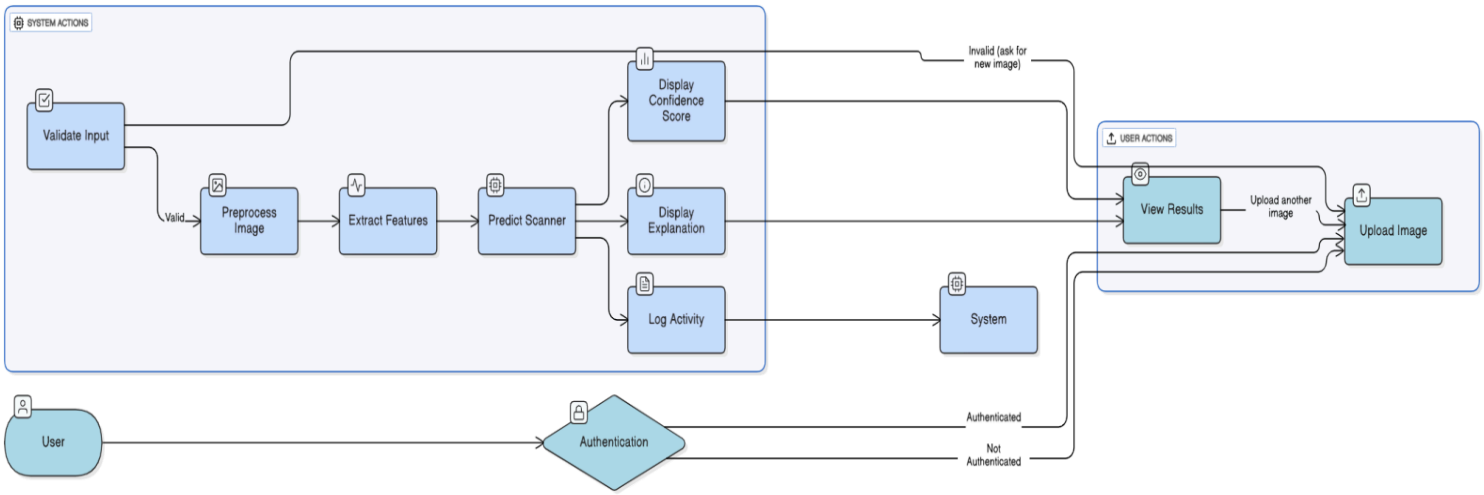
The design phase resulted in a well-structured system that balances technical depth with user accessibility. Each module is independently testable, and the overall pipeline supports forensic workflows with traceability and interpretability.

3.2.1 Data flow diagram



3.2.2 UML Diagrams





eraser

eraser

3.3 Methodology

The methodology adopted for this project follows a structured, stepwise approach to build a robust forensic scanner identification system using machine learning. It ensures reproducibility, scalability, and interpretability across diverse image formats and environments.

Step 1: Problem Definition

- Identify the challenge of attributing scanned images to their source scanners.
- Define objectives: feature extraction, classification, reproducibility, and dashboard deployment.

Step 2: Data Collection and Preparation

- Gather ~4568 scanned images from 11 scanner classes.
- Supported formats: TIFF, PNG, JPEG, TFF, PDF.
- Resolutions: 150, 300, and 600 DPI.
- Split: 80% training, 10% validation, 10% testing.
- Apply preprocessing: grayscale conversion, resizing, denoising, normalization.

Step 3: Feature Extraction

- Extract scanner-specific patterns using:
- PRNU and SPN: Sensor-level noise
- FFT: Frequency-domain signatures
- Wavelet Transforms: Multi-scale texture analysis
- Statistical Descriptors: Mean, variance, skewness, kurtosis
- Validate feature shapes using diagnostic scripts.

Step 4: Model Development

- Implement and compare three classifiers:
- CNN: Deep learning model for spatial patterns
- Random Forest: Ensemble model for statistical features
- SVM: Kernel-based model for high-dimensional vectors
- Train models using 80% of data, validate on 10%, test on 10%.
- Tune hyperparameters using grid search and apply regularization.

Step 5: Evaluation

- Use metrics: Accuracy, Precision, Recall, F1-Score, Confusion Matrix.
- Compare model performance across scanner classes.
- Analyze misclassifications and edge cases.

Step 6: Deployment

- Build a Streamlit dashboard for real-time interaction.
- Features:
- Image upload
- Prediction display with confidence score
- Visualization of feature maps
- Logging and fallback handling

Step 7: Documentation and Reproducibility

- Maintain clean folder structure and versioned scripts.
- Log all preprocessing, training, and prediction steps.
- Ensure reproducibility across Windows and Google Colab environments.

CHAPTER 4

SYSTEM IMPLEMENTATION

The implementation phase involved translating the system design into a working forensic scanner identification pipeline. Each module was developed, tested, and integrated to ensure end-to-end functionality, reproducibility, and user accessibility.

4.1 Development Environment

- **Local Setup**: Windows 11,VS Code with Python 3.10.11, Anaconda, and Jupyter Notebook
- **Cloud Setup**: Google Colab (GPU-enabled) for large-scale training and testing
- **Libraries Used**: NumPy, OpenCV, pywt, SciPy, TensorFlow, Keras, Pandas, tqdm, Streamlit

4.2 Module Implementation

1. Input Validation

- Checks image format (TIFF, PNG, JPEG, PDF, TFF)
- Verifies resolution (150, 300, 600 DPI)
- Logs file metadata and errors

2. Preprocessing

- Grayscale conversion using OpenCV
- Resizing to uniform dimensions
- Denoising via Gaussian filters
- Normalization for consistent feature scaling

3. Feature Extraction

- PRNU/SPN: Sensor noise patterns

- FFT: Frequency-domain analysis
- Wavelet Transforms: Multi-scale texture extraction
- Statistical Features: Mean, variance, skewness, kurtosis
- Features stored as files for reuse and validation

4. Model Training

- Implemented three classifiers:
- CNN: Trained on image patches and feature maps
- Random Forest: Trained on statistical vectors
- SVM: Trained on normalized wavelet features
- Used early stopping, dropout, and grid search for optimization
- Training logs and metrics saved for reproducibility

5. Evaluation

- Metrics: Accuracy, Precision, Recall, F1-Score, Confusion Matrix
- Class-wise performance analyzed
- Misclassified samples logged for review

6. Deployment

- Streamlit Dashboard:
- Upload scanned image
- View predicted scanner class and confidence score
- Display feature maps and logs
- Handle unsupported formats with fallback messages
- Dashboard tested on both local and cloud environments

CHAPTER 5

RESULTS AND DISCUSSIONS

This chapter presents the performance outcomes of the AI TraceFinder system, highlighting model accuracy, class-wise metrics, explainability, and forensic relevance. The results validate the system's ability to identify scanner sources with high precision and interpretability.

5.1 Model Performance Overview

The hybrid model combines **CNN-learned visual patterns** with **27 handcrafted statistical features**, enabling robust classification across 11 scanner classes.

- **Training Set**: 3,654 scans
- **Test Set**: 914 scans
- **Image Shape**: (256×256 grayscale)
- **Feature Vector**: 27 features per image
- **Test Accuracy**: **93.65%**

5.2 Class-wise Evaluation

```
Test Accuracy: 93.65%

Classification Report:
              precision    recall  f1-score   support

   Canon120-1       0.91      0.89      0.90        83
   Canon120-2       0.84      0.83      0.84        83
     Canon220       0.89      0.92      0.90        83
  Canon9000-1       0.95      0.87      0.91        83
  Canon9000-2       0.88      0.95      0.91        82
  EpsonV370-1       0.99      0.95      0.97        83
  EpsonV370-2       0.95      0.99      0.97        84
   EpsonV39-1       0.94      0.96      0.95        83
   EpsonV39-2       0.96      0.95      0.96        84
    EpsonV550       1.00      0.99      0.99        83
           HP       0.99      1.00      0.99        83

     accuracy                           0.94       914
    macro avg       0.94      0.94      0.94       914
 weighted avg       0.94      0.94      0.94       914
```

5.3 Confusion Matrix Insights

- Most misclassifications occurred between scanners of the same brand or resolution.
- False positives were limited and often involved adjacent classes.
- Mitigation: Deeper CNN layers and DPI normalization improved separation

5.4 Explainability Tools

To enhance forensic interpretability, the system integrates:

- Grad-CAM: Highlights image regions influencing CNN predictions
- SHAP: Visualizes feature importance in ML models (e.g., Random Forest)

These tools provide visual justification for predictions, supporting legal and forensic use cases.

5.5 Dashboard Feedback

The Streamlit dashboard was tested by peers and mentors:

- **Strengths**:
    - Fast predictions
    - Clear confidence scores
    - Visual feedback and logs
- **Suggestions**:
- Add batch upload
- Include Grad-CAM overlays in UI

5.6 Forensic Impact

The system supports multiple forensic applications:

- Source Attribution: Identifies scanner origin of a document
- Fraud Detection: Flags unauthorized or tampered scans
- Legal Verification: Validates authenticity of scanned evidence
- Explainability: Provides visual proof for classification decisions

5.7 Summary

- The hybrid model achieved **93.65%** accuracy across diverse scanner brands
- Feature fusion (CNN + handcrafted) improved robustness
- Explainability tools enhanced trust and forensic usability
- The system is reproducible, scalable, and ready for real-world deployment

CHAPTER 6

CONCLUSION AND FUTURE WORK

The AI TraceFinder system successfully demonstrates the feasibility of forensic scanner identification using a hybrid machine learning approach. By combining CNN-based visual pattern recognition with handcrafted statistical features, the system achieved a high test accuracy of 93.65% across 11 scanner classes.

Key accomplishments include:

- Development of a reproducible, modular pipeline for scanner attribution
- Integration of robust preprocessing and feature extraction techniques (FFT, PRNU, statistical descriptors)
- Comparative evaluation of multiple classifiers (CNN, Random Forest, SVM)
- Deployment of an interactive Streamlit dashboard for real-time predictions
- Incorporation of explainability tools (Grad-CAM, SHAP) to enhance forensic interpretability

The system is scalable, interpretable, and applicable to real-world forensic and legal scenarios involving document authenticity and source verification.

6.2 Future Work

While the current system performs well, several enhancements can further improve its accuracy, scalability, and forensic utility:

- Expand Dataset: Include more scanner brands, models, and resolutions to improve generalization
- Batch Processing: Enable multi-image upload and batch classification in the dashboard
- Advanced Explainability: Integrate Grad-CAM overlays directly into the UI for visual justification

- Noise Simulation: Introduce synthetic noise/tampering to test robustness against adversarial inputs
- Mobile Deployment: Explore lightweight model variants for mobile or edge deployment
- Legal Integration: Align output formats with digital evidence standards for courtroom admissibility
- AutoML Integration: Use automated model selection and hyperparameter tuning for faster experimentation

This project lays a strong foundation for future research in scanner forensics and digital document authentication. With further refinement, AI TraceFinder can evolve into a powerful tool for law enforcement, legal professionals, and forensic analysts.

CHAPTER 7

APPENDICES

7.1 Source Code

The complete source code for the AI TraceFinder forensic scanner identification system is available on GitHub:

- Repository: https://github.com/amitkushwaha001/AI_TraceFinder.git
- Main Modules:
  - `preprocessing.py`: Image resizing, grayscale conversion, denoising
  - `feature_extraction.py`: FFT, wavelet, PRNU, statistical descriptors
  - `models.py`: CNN architecture, Random Forest, SVM classifiers
  - `train.py`: Model training and evaluation pipeline
  - `streamlit_app.py`: Dashboard deployment and user interaction
  - `utils/`: Diagnostic scripts, shape audits, logging utilities
- Requirements: Python 3.10.11, dependencies listed in `requirements.txt`
- Execution:

```
streamlit run streamlit_app.py
```

7.2 Learning Experiences

During this internship, I gained hands-on experience in:

- Designing reproducible ML pipelines for forensic applications
- Implementing feature extraction techniques (FFT, PRNU, wavelets)
- Comparing classical ML models with deep learning architectures
- Building and deploying interactive dashboards using Streamlit
- Applying explainability tools (Grad-CAM, SHAP) for model transparency

- Managing large datasets in both local and cloud environments
- Writing clean, modular code with proper logging and validation
- Understanding the importance of traceability and interpretability in forensic systems

This project strengthened my skills in Python, machine learning, and user-centric product development, while deepening my understanding of digital forensics.

7.3 SWOT Analysis

Strengths	Weaknesses
High model accuracy (93.65%)	Limited scanner diversity in dataset
Hybrid feature fusion (CNN + handcrafted)	No batch upload in dashboard
Explainability tools integrated	Grad-CAM not yet visualized in UI
Reproducible across platforms	Requires manual dataset setup

Opportunities	Threats
Expand to mobile or edge deployment	Adversarial tampering of scanned images
Legal integration for courtroom use	Dataset bias or class imbalance
AutoML for faster experimentation	Hardware similarity across scanners may confuse model