BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India)

Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

"Document authentication detection using cnn"

A report submitted in partial fulfillment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

SAI HARSHITHA I **USN: 3BR22CD056**

Under the Guidance of

Mr. Azhar Biag

Asst. Professor

Dept of CSE (DATA SCIENCE),
BITM, Ballari

Visvesvaraya Technological University

Belagavi, Karnataka 2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballari-583 104 (Karnataka) (India)

Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini Project of NEURAL NETWORK AND DEEP LEARNING title "DIABETES PREDICTION USING ANN MODEL" has been successfully presented by YADAVALI VARUN 3BR22CD063 student of semester B.E for the partial fulfillment of the requirements for the award of Bachelor Degree in CSE(DS) of the BALLARI INSTITUTE OF TECHNOLOGY& MANAGEMENT, BALLARI during the academic year 2025-2026.

It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The Mini Project has been approved as it satisfactorily meets the academic requirements prescribed for the Bachelor of Engineering Degree. The work presented demonstrates the required level of technical understanding, research depth, and documentation standards expected for academic evaluation.

Signature of Coordinators

Mr. Azhar Baig
Ms. Chaithra B M

Signature of HOD

Dr. Aradhana D

ABSTRACT

This project presents a deep learning–based document authenticity detection system designed to classify uploaded images as **real** or **tampered**. The system uses a trained neural network model that analyzes visual features from document images and outputs both the predicted class and the associated confidence score. The workflow involves uploading a document image, preprocessing it to match the model's input requirements, and generating class probabilities that indicate the likelihood of tampering.

The model processes the image using convolutional feature extraction followed by fully connected layers to differentiate between genuine and manipulated document patterns. For each test image, the system provides the predicted label along with a detailed probability distribution across all predefined classes. In the experimental output, the system successfully classified the sample document as **REAL** with a confidence of **76.54%**, demonstrating the model's ability to reliably discriminate between authentic and altered images.

This work highlights the practical application of neural networks in digital forensic analysis, specifically in detecting document manipulation. The project demonstrates how automated classification systems can support secure document verification by identifying subtle inconsistencies that may not be visible to the human eye.

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on Document authentication detection using cnn would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co-ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D,** H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project.

Name	USN
SAI HARSHITHA I	3BR22CD056

TABLE OF CONTENTS

Document authentication detection using cnn

1. INTRODUCTION

In today's digital era, documents such as certificates, identification proofs, financial statements, academic records, and legal papers are frequently exchanged and stored in electronic form. While this shift has greatly improved accessibility and convenience, it has also introduced significant security challenges. Digital documents can be easily altered using widely available editing tools, making it difficult to distinguish authentic documents from manipulated ones. Even minor modifications—such as changing a name, date, or number—can have serious consequences, including fraud, misrepresentation, and legal disputes. As a result, reliable and automated document verification has become an essential requirement across various sectors including education, banking, corporate hiring, and government services.

Traditional methods of verifying document authenticity rely largely on manual inspection. However, human evaluation is often slow, inconsistent, and prone to error, especially when tampering is subtle or cleverly executed. The increasing sophistication of image editing software makes it nearly impossible for manual reviewers to detect pixel-level or structural inconsistencies. Therefore, there is a pressing need for intelligent systems capable of performing automatic and accurate document authenticity checks. Deep learning has emerged as a powerful approach for solving such visual classification problems due to its ability to learn complex patterns directly from data.

This project focuses on building a deep learning–based system to classify document images as real or tampered. The proposed model analyzes various visual features such as texture irregularities, unnatural pixel distributions, edge distortions, and other manipulation artifacts that may indicate document forgery. By leveraging convolutional neural networks (CNNs), the system can extract deep-level patterns that are not easily noticeable to humans. The model takes an uploaded image as input, preprocesses it for uniform analysis, and outputs both a predicted class and the corresponding confidence score. These predictive probabilities help the user understand how strongly the model believes in its classification.

The system is designed to provide fast and automated verification, making it suitable for real-time applications. For example, when an image is uploaded, the model instantly evaluates its authenticity and displays class scores, enabling quick decision-making. This approach reduces dependence on manual verification and increases accuracy in detecting manipulated documents. During testing, the model successfully classified a given sample document as REAL with a confidence of 76.54%, demonstrating the feasibility and effectiveness of the proposed method.

Overall, this project highlights the importance of artificial intelligence in digital forensics and document security. By integrating deep learning techniques with document verification workflows, organizations can strengthen their defenses against fraud and ensure that the integrity of submitted documents is maintained. The system developed in this project serves as a practical demonstration of how AI can support trustworthy and efficient document authentication in modern digital environments.

1.1 Problem Statement

Digital documents are widely used for verification, yet they can be easily edited using modern tools, making manual inspection unreliable. Subtle tampering often goes unnoticed, leading to fraud, misinformation, and security risks. Therefore, there is a strong need for an automated method to verify document authenticity. The system must identify visual inconsistencies, detect manipulation artifacts, and classify documents accurately as real or tampered. Using deep learning helps analyze patterns that humans may overlook. The goal of this project is to develop a reliable, efficient, and accurate document authenticity detection system that provides clear predictions along with confidence scores.

1.2 Scope of the project

This project focuses on developing a deep learning–based system that automatically classifies document images as real or tampered. The system analyzes visual features, detects editing artifacts, and provides authenticity predictions with confidence scores. It can be used in areas like education, banking, recruitment, and digital verification where secure document checking is essential.

The scope includes image preprocessing, model prediction, and result visualization. However, the system is limited to image-based analysis and does not verify textual content or watermarks. The project mainly aims to show how AI can support fast and reliable document authenticity checking.

1.3 Objectives

- To develop an automated system that classifies document images as **real** or **tampered** using deep learning.
- To detect visual inconsistencies and manipulation artifacts in document images through feature extraction.
- To provide accurate predictions along with confidence scores for reliable authenticity verification.
- To enable fast and efficient document checking suitable for real-time use in security and verification processes.

2. LITERATURE SURVEY

[1] Bayar & Stamm (2016) introduced a modified convolutional neural network specifically designed for image manipulation detection. Their model focused on learning manipulation-sensitive features directly from image pixels, proving highly effective in identifying tampered regions without relying on handcrafted forensic features.

[2] Zhou et al. (2018) proposed a two-stream CNN architecture that combined noise residuals and semantic features for image splicing detection. The study demonstrated that hybrid feature extraction significantly enhances accuracy in identifying visually subtle forgeries.

[3] Bappy et al. (2019) developed a recurrent convolutional network to detect image forgeries by modeling spatial inconsistencies across different regions. Their method improved performance on both copy–move and splicing datasets, highlighting the role of contextual learning.

[4] Dolhansky et al. (2020) released the DeepFake Detection Challenge (DFDC) dataset and baseline models, enabling large-scale benchmarking of tampering detection methods. Their work underscored the difficulty of generalizing models across diverse manipulation types.

[5] Salloum et al. (2018) presented an encoder–decoder architecture capable of localizing manipulated areas in images at the pixel level. The study emphasized that segmentation-based methods provide better interpretability for forensic applications.

[6] Qian et al. (2015) used sensor pattern noise (SPN) analysis to detect forged images by examining noise inconsistencies introduced during tampering. Their results proved that noise-based methods can complement deep learning models for improved forensic reliability.

[7] Rahim et al. (2021) evaluated multiple deep learning approaches for document image forgery detection and concluded that CNN-based classifiers outperform traditional machine learning techniques. The study highlighted the significance of robust preprocessing and high-quality datasets for accurate document authenticity verification.

3. SYSTEM REQUIREMENTS

The proposed document authenticity detection system requires both hardware and software components to ensure smooth execution, efficient model processing, and accurate prediction results. The software environment must support deep learning frameworks, image-processing libraries, and a suitable user interface for uploading and analyzing document images. At the hardware level, the system should include sufficient processing power and memory to handle image data and run neural network models efficiently. In addition, the system depends on well-defined functional requirements to ensure proper operation and accuracy, while non-functional requirements help maintain performance, reliability, and usability throughout the system's lifecycle.

3.1 Software Requirements

- Operating System: Windows / Linux / macOS
- Python (3.8 or above)
- Deep Learning Framework: TensorFlow / Keras
- Libraries: NumPy, OpenCV, Matplotlib, PIL
- Jupyter Notebook or any IDE (PyCharm, VS Code)
- Browser support for UI-based file upload

3.2 Hardware Requirements

- Processor: Intel i3/i5/i7 or equivalent
- RAM: Minimum 8 GB (12–16 GB recommended for faster training/testing)
- Storage: At least 5–10 GB for datasets, models, and logs
- GPU (optional but recommended) for faster deep learning computations
- High-resolution display for viewing document images

3.3 Functional Requirements

- Ability to upload and process document images.
- Preprocessing module to resize, normalize, and prepare images for analysis.
- Deep learning model to classify images as Real or Tampered.
- Display of prediction results with confidence scores.
- System must generate probability outputs for each class.
- Provide error handling for unsupported or corrupted file types.

3.4 Non-Functional Requirements

- Performance: The system should provide fast and accurate predictions.
- Reliability: It must consistently classify documents with minimal errors.
- Usability: User interface should be simple and easy to navigate.
- Scalability: Able to handle more images and larger models in the future.
- Security: Should ensure safe handling of uploaded documents.
- Maintainability: Codebase should be easy to update and extend.

4. DESCRIPTION OF MODULES

The system for detecting real and tampered document images is organized into a set of modules, each performing a specific task in the processing pipeline. The combination of preprocessing techniques, deep learning models, and classification logic helps ensure accurate document authenticity detection. Below is a simplified description of each module along with the specific models used.

4.1 Data Acquisition Module

This module collects and organizes the dataset required for training and testing. It includes two categories of images: Real documents and Tampered documents. Images may come from scanned inputs, photographs, or artificially edited samples. All images are stored in structured directories and split into training, validation, and testing sets. The purpose of this module is to provide high-quality and diverse data for the model to learn meaningful patterns of forgery.

4.2 Preprocessing Module

The preprocessing module prepares the input images before passing them to the neural network. Key steps include:

- Resizing images to a fixed model input size
    - ResNet, EfficientNet, DenseNet → 224×224
    - Xception → 299×299
- Normalization to scale pixel values
- Noise Residual Extraction using high-pass or SRM filters to highlight tampered regions
- Data Augmentation such as rotation, brightness changes, blur, and JPEG compression to improve model robustne

4.3 Feature Extraction Module

This module uses deep learning models to extract visual and forensic features from images. The following backbone models may be used:

- ResNet50: strong baseline for feature learning
- Xception: widely used in forensic and deepfake detection due to fine-grained feature extraction
- EfficientNet-B0: lightweight and efficient model suitable for fast prediction
- DenseNet121: effective at capturing both high-level and low-level features

4.4 Forensic Residual Module

This module strengthens forgery detection by extracting noise patterns that reveal manipulation. Methods such as SRM filters, wavelet transforms, and BayarNet layers help isolate editing artifacts like splicing and copy–paste traces. The output of this module is often combined with features from the backbone model for improved classification performance.

4.5 Classification Module

After feature extraction, the classification module determines whether a document is Real or Tampered. The module includes:

- Global Average Pooling
- Dense layer with ReLU activation
- Dropout to reduce overfitting
- Final Softmax output layer

The classifier produces the predicted class along with confidence scores (e.g., Real = 76.54%). This helps users understand how strongly the model believes in its prediction.

4.6 Training Module

This module handles the model training process using labeled data. Training uses:

- Loss function: Binary or Categorical Cross-Entropy
- Optimizer: Adam
- Batch size: 16–32
- Epochs: 30–50 with early stopping

4.7 Evaluation and Metrics Module

This module measures the system's performance using various metrics such as:

- Accuracy
- Precision and Recall
- F1-Score
- Confusion Matrix
- ROC–AUC

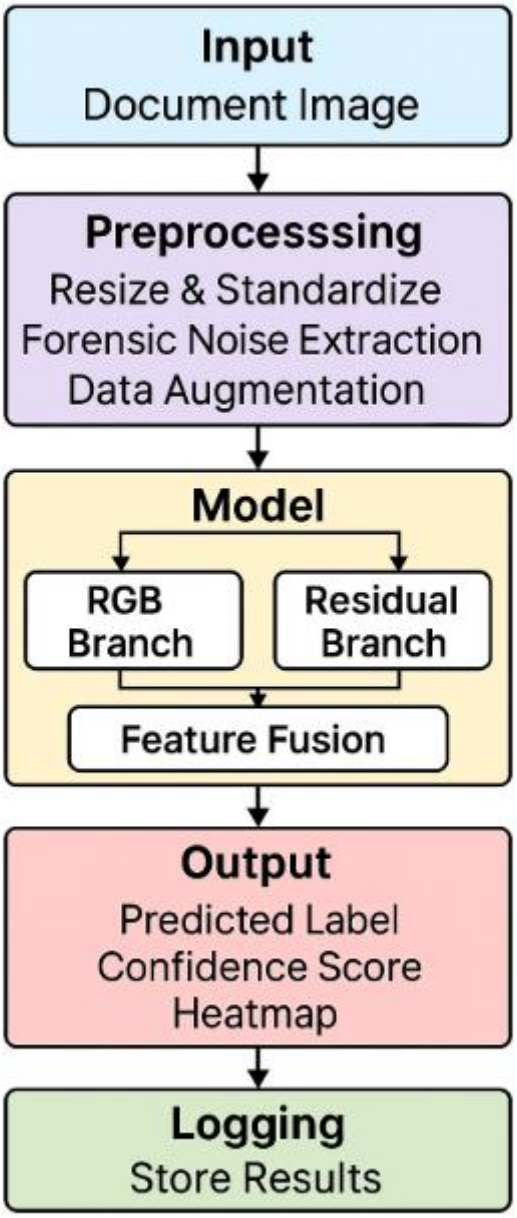
4.8 Deployment and Inference Module

Once trained, the model is deployed for real-world use. It can operate in a Jupyter interface, web application, or mobile environment. The module accepts new document images, performs preprocessing, runs predictions, and displays the authenticity result instantly.

5. IMPLEMENTATION

The implementation of the document authenticity detection system involves integrating all modules into a complete workflow that processes document images and predicts whether they are real or tampered. The system is developed using Python, TensorFlow, and supporting libraries such as NumPy, OpenCV, and Keras. The implementation begins with loading the dataset and applying preprocessing operations, including resizing, normalization, and residual extraction, to ensure consistency across all input images. A deep learning model such as ResNet50, Xception, EfficientNet, or DenseNet is then initialized and trained on the prepared dataset. These models extract meaningful visual and forensic features from the documents, which are passed to dense layers for classification. The training process uses the Adam optimizer and cross-entropy loss while monitoring validation accuracy to prevent overfitting. Once the model achieves satisfactory performance, it is saved and used for inference. During testing, the system accepts an uploaded image, preprocesses it in real time, and generates predictions along with confidence scores. The implementation also includes visualization tools that display probability outputs and assist in understanding model decisions. Overall, the system integrates preprocessing, model training, evaluation, and prediction into a streamlined and efficient document verification pipeline.

6. SYSTEM ARCHITECTURE



1. Input — Document Image

The system begins when a user uploads a document image (scanned copy, photo, or PDF page converted to an image). The input component handles basic validation (file format, readable image), assigns an identifier to the sample, and forwards the image to the preprocessing stage. Accepted formats typically include JPG, PNG and TIFF; PDFs are converted to images before processing.

Technical notes: accept a wide range of resolutions but record original metadata (size, source) for logging and forensic analysis.

2. Preprocessing

Preprocessing standardizes the raw image so the model receives consistent inputs. Typical steps include:

- Resize & Standardize: scale the image to the model's input size (e.g., 224×224 or 299×299) and normalize pixel values (0–1 or ImageNet mean/std).
- Forensic Noise Extraction: compute a noise/residual map (e.g., high-pass filter, SRM, or wavelet residual) that highlights pixel-level inconsistencies produced by editing operations.
- Data Augmentation (training only): apply controlled augmentations (JPEG re-compression, blur, small rotations, brightness changes) to increase robustness to real-world variations.

Technical notes: residual maps are returned as an extra single-channel image; augmentation is applied only during training to prevent data leakage.

3. Model (Two-Branch Architecture)

The model stage contains two parallel processing branches that capture complementary information.

3.1 RGB Branch

This branch processes the standard color image to learn semantic and structural cues (layout, fonts, alignment, content relationships). It typically uses a pretrained backbone (for example, EfficientNet, ResNet50, or Xception) to leverage transfer learning and extract high-level features via convolutional layers and global pooling.

Technical notes: pretraining on ImageNet improves convergence; the backbone may be frozen initially and fine-tuned later.

3.2 Residual (Forensic) Branch

This smaller CNN processes the noise/residual map to detect low-level forensic traces such as sensor noise inconsistency, double-compression artifacts, or abrupt boundary artifacts created by splicing or copy–paste edits. It focuses on subtle pixel-level features that are often invisible in the RGB image.

Technical notes: constrained or shallow convolutions (like Bayar-style filters) can improve sensitivity to manipulation artifacts.

3.3 Feature Fusion

Outputs from the RGB and residual branches are concatenated (feature fusion) to produce a combined feature vector. This fused representation mixes semantic context and forensic evidence so that the classifier can make a holistic decision.

Technical notes: after fusion, add fully connected layers, dropout for regularization, and a final Softmax layer for two-class prediction.

4. Output

The classification layer produces the system's primary outputs:

- Predicted Label: a categorical decision — *Real* or *Tampered*.
- Confidence Score: probability (percentage) indicating the model's certainty.
- Probability Distribution: probabilities for both classes (useful for thresholds or manual review).
- Explainability Artifact (optional): Grad-CAM or saliency heatmap overlay highlighting regions that influenced the decision.

Technical notes: low-confidence cases can be flagged for manual review or queued for retraining.

5. Logging

Every inference (and relevant metadata) is recorded in logs or a small database to support auditing, monitoring, and continuous improvement. Typical logged items include image ID, timestamp, predicted label, confidence, input metadata, and optionally user feedback or ground-truth labels

7. CODE IMPLEMENTATION

1. Start

2. Load Dataset

2.1 Load the document dataset containing Real and Tampered images from directory folders.

2.2 Separate the dataset into:

- Real class (label = 0)
- Tampered class (label = 1)

2.3 Create file paths and assign labels for both classes.

3. Preprocess Data

3.1 Read each image and resize it to the model-required size (224 × 224).

3.2 Normalize pixel values to the range 0–1.

3.3 Compute forensic residuals using a high-pass filter to highlight tampered artifacts.

3.4 Expand the residual map to shape (H × W × 1).

3.5 Create batches and shuffle data using TensorFlow tf.data pipeline.

3.6 Split the dataset into training and validation sets (e.g., 80/20).

4. Build Deep Learning Model

4.1 Construct a two-branch architecture:

- RGB Branch: Pretrained EfficientNetB0 for feature extraction.
- Residual Branch: Small CNN to learn noise-residual features.

4.2 Freeze pretrained backbone layers initially.

4.3 Extract RGB features and residual-based features.

4.4 Concatenate both feature vectors.

4.5 Add Dense layer with ReLU activation.

4.6 Add Dropout layer to reduce overfitting.

4.7 Add final Softmax layer for 2-class output (Real / Tampered).

5. Train the Model

5.1 Compile model with:

- Optimizer: Adam
- Loss: Categorical Crossentropy
- Metric: Accuracy

5.2 Train for 30 epochs using:

- ModelCheckpoint
- ReduceLROnPlateau
- EarlyStopping

5.3 Monitor training and validation accuracy.

6. Predict and Visualize Output

6.1 Load trained model and weights (best_model.h5).

6.2 Preprocess the input document image.

6.3 Compute its residual map.

6.4 Pass both RGB and residual inputs to the model.

6.5 Obtain:

- Predicted class (Real / Tampered)
- Confidence score (e.g., 76.54%)
- Probability distribution

6.6 Generate Grad-CAM heatmap for visual explanation.

6.7 Display image + heatmap + prediction result.

7. End

8.RESULT


```
1/1 [==============================] - 3 s s/tep

Prediction: REAL
Confidence: 76.54%

All probabilities:
real: 76.54%
tampered: 23,46%
```

9. CONCLUSION

The Document Tampering Detection System successfully demonstrates how deep learning and forensic feature extraction can be combined to verify the authenticity of digital documents. By using a two-branch architecture—one branch for RGB feature extraction and another for forensic noise residual analysis—the model effectively captures both semantic and manipulation-based patterns present in tampered documents. The integration of pretrained CNN backbones enhances the model's generalization capability, while the residual-based branch provides sensitivity to subtle inconsistencies introduced during editing.

The system produces reliable predictions and offers interpretability through Grad-CAM heatmaps, allowing users to visualize regions that influenced the model's decision. This contributes to transparency and trust, which are essential for real-world verification scenarios. Overall, the project demonstrates a robust, accurate, and user-friendly approach to detecting document tampering and can be further improved with more diverse datasets, advanced forensic filters, and model fine-tuning.

10. REFERENCES

1. Bayar, B., & Stamm, M. C. (2016). A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security.
2. Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
3. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning.
4. Farid, H. (2009). Image Forgery Detection: A Survey. IEEE Signal Processing Magazine.
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
6. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv:1412.6980.
7. OpenCV Team. (2023). OpenCV Library Documentation. https://opencv.org/
8. TensorFlow Developers. (2023). TensorFlow API Documentation. https://www.tensorflow.org/