The Development of an Image Processing System for Document Forgery Detection Using Python Programming Language

ADEYEMO Isiaka Akinkunmi

Department of Computer Science, Faculty of Computing and Informatics (FCI), Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria.

CHAPTER ONE

INTRODUCTION

Digital technology has completely transformed how we create, share, and store documents. While this has made life easier in many ways, it's also opened the door to new challenges—one of the biggest being document forgery. From financial institutions and educational systems to legal processes and business operations, forged documents can cause serious problems. They can lead to financial losses, compromise security, and even damage reputations. Because of this, finding better ways to detect forged documents has become more important than ever.

Traditionally, identifying forged documents has relied on manual inspections. But let's face it—this approach is slow, prone to errors, and just doesn't scale well. As forgery techniques get more advanced, we need smarter solutions that can keep up. That's where automated systems come in, offering the potential for more accurate and efficient detection.

Image processing, a fascinating area of computer science, plays a key role here. It focuses on analyzing and manipulating digital images to extract meaningful insights. Techniques like edge detection, texture analysis, and feature extraction can reveal subtle changes in documents that would otherwise go unnoticed. When combined with machine learning, these systems can even learn from data and improve their accuracy over time.

This project aims to develop an image processing system to detect document forgeries using Python. Python is a fantastic choice for this, thanks to its powerful libraries and frameworks for image processing and machine learning. The system will analyze visual patterns and anomalies in documents, making it easier to identify forged ones. By automating this process, we're looking to create a solution that's both scalable and reliable.

The project will dive into existing methods for forgery detection, apply advanced image processing techniques, and evaluate the system's performance using practical metrics. The ultimate goal is to develop a tool that's not only effective but also accessible—helping to tackle the growing issue of document forgery in today's digital world.

Background of the study

Document forgery, the act of altering or faking documents with the intent to deceive, has become a pressing issue across various sectors such as finance, legal, healthcare, and education. In the digital age, the manipulation of documents has become easier, and many organizations now rely on digital records, making it increasingly difficult to distinguish between genuine and fraudulent documents. The growing use of digital tools to alter documents, such as word processors and image editing software, has outpaced traditional methods of verification. As more documents go digital, the risk of them being altered or faked increases, making it harder to trust their authenticity. The usual methods of checking documents for tampering just aren't cutting it anymore, especially as technology improves.

For example, in the financial sector, forged checks, invoices, and contracts can result in significant monetary losses. Similarly, in the legal system, fake identification documents or altered contracts can undermine the entire legal process. In education, the circulation of forged certificates and transcripts threatens the integrity of

academic qualifications, causing widespread trust issues. These types of fraud are not only damaging to the individuals and organizations involved but also contribute to a broader societal problem, reducing confidence in the authenticity of important documents.

That's where image processing comes in. With the right technology, we can automatically scan and analyze documents to spot signs of forgery, saving time and reducing errors. Python, a powerful programming language, offers a variety of tools that make it easier to build such systems. By using these tools, we can develop solutions that quickly and accurately detect document fraud, keeping important records secure.

In response to these challenges, there is a growing demand for more reliable and efficient methods of detecting document forgery. Traditional techniques like manual inspection or using watermarks are not sufficient to identify more sophisticated forgeries. This has led to the exploration of advanced technological solutions, particularly in the field of image processing.

Image processing technology offers powerful tools to analyze the visual elements of a document. By identifying inconsistencies or anomalies in the document's structure, fonts, images, and layout, an automated system can determine whether a document has been tampered with. This can significantly reduce the time and effort needed for verification, especially for organizations that handle a large volume of documents regularly.

Image processing technology has emerged as a promising solution to this problem. By using algorithms to analyze the pixel-level information in digital documents, image processing systems can detect subtle signs of tampering, such as changes in font styles, distortions in images, or mismatched patterns in text. Techniques like optical character recognition (OCR), feature extraction, and pattern recognition allow systems to scrutinize documents in ways that go beyond simple visual inspection. These systems can even identify inconsistencies in the background of documents or highlight irregularities in the document's layout that may not be immediately obvious.

Python, a widely-used programming language, has become one of the preferred tools for developing image processing systems. Python's simplicity, combined with powerful libraries such as OpenCV, PIL (Pillow), and NumPy, makes it an ideal choice for both beginners and experienced developers. These libraries offer robust functionalities for image manipulation and analysis, from basic image transformations to complex feature detection. Python also supports machine learning frameworks like TensorFlow and Keras, which can be used to train models for forgery detection, improving the accuracy and efficiency of the system over time.

The development of an image processing system for document forgery detection is not just a technical challenge; it is a necessary step toward securing digital records in an increasingly digital world. Such a system would not only streamline the verification process for individuals and organizations but also help in reducing fraud, safeguarding legal and financial transactions, and ensuring the credibility of academic institutions.

Statement of the study

Document forgery is one of the most prevalent forms of fraud in today's digital world. As technology advances, so does the sophistication of document manipulation techniques. What was one a problem restricted to physical forgeries—such as altered signatures or tampered paper documents—has now become an urgent digital issue. With the rise of image-editing software and digital tools, it is increasingly difficult to detect forged documents simply by visual inspection. These manipulated documents are often so convincing that they can pass for legitimate, creating a significant risk in sectors that rely on the authenticity of their records, such as banking, legal affairs, government, healthcare, and education.

The consequences of document forgery can be catastrophic. For example, in the financial sector, forged checks, loan documents, and contracts can lead to significant monetary losses. In the legal system, the presentation of falsified evidence or fraudulent contracts can result in wrongful convictions, legal disputes, or the collapse of cases. Similarly, in education, the circulation of fake degrees, certificates, and transcripts undermines the credibility of institutions, devalues academic qualifications, and promotes dishonest practices.

Despite the growing prevalence of document forgery, many organizations still rely on outdated methods to detect fraud. The traditional approach to document verification often involves manual inspection or simple software tools that offer limited functionality. Manual inspection is highly dependent on human judgment and expertise, making it prone to errors, fatigue, and inconsistencies. For instance, human inspectors may miss subtle signs of tampering, such as slight variations in font type or pixel manipulation, which are often difficult to detect without specialized tools.

While some digital forensics tools exist, they are generally inadequate for handling the volume, variety, and complexity of documents encountered in everyday transactions. Many current systems focus on detecting basic forms of tampering, like signature verification or watermark checks, but these techniques can easily be bypassed by advanced forgers using high-end software. Furthermore, many existing tools are expensive, complicated to use, and often require a certain level of expertise, limiting their accessibility to smaller organizations or individuals who need them the most.

Additionally, many forgery detection systems fail to provide accurate results when faced with minor alterations or highly sophisticated forgery methods. Subtle changes, such as small alterations to text or images that blend seamlessly into the document, can easily go unnoticed by both manual inspectors and existing digital systems. This is particularly problematic in industries where the integrity of a document is paramount and even small discrepancies can have significant consequences.

The problem is compounded by the sheer scale of documents that need to be verified. In sectors like education, finance, and law, the volume of documents being processed on a daily basis is staggering. Manually checking each document for authenticity is not only time-consuming but also inefficient. As organizations continue to embrace digital transformation, the need for faster, more efficient, and more reliable methods of forgery detection becomes even more critical.

Therefore, this study seeks to address the pressing need for an automated, efficient, and accurate solution to detect document forgery. By leveraging image processing techniques and the power of Python, the goal is to develop a system capable of analyzing various types of documents, identifying signs of manipulation, and providing an automated, reliable method for forgery detection. Such a system could significantly reduce the time, effort, and human error involved in document verification, helping to mitigate the risks associated with document fraud and ensuring the security of digital records.

Aim and objectives

Aim:

The aim of this project is to develop an image processing system for document forgery detection using python programming language.

Objectives:

The specific objectives are to;

i. To review existing document forgery detection methods and identify their strengths and limitations.
ii. To explore image processing techniques suitable for detecting document manipulation.
iii. To design and implement a Python-based system capable of identifying forged documents.
iv. To suggest potential improvements and future directions for forgery detection research.

Significance of the Study

The significance of this study is rooted in its ability to tackle a growing issue that affects almost every industry today: document forgery. In an era where digital documents are becoming the norm in sectors like finance, law, education, and government, the risk of fraudulent activities is also on the rise. Whether its fake degrees being passed off as legitimate qualifications, altered contracts changing the terms of agreements, or

manipulated financial documents, the consequences of document forgery can be severe. That's where this study comes in—by developing an automated image processing system for forgery detection, we're aiming to bring a practical, efficient solution to this pressing problem.

1. **Enhancing Security and Preventing Fraud:**

   At the heart of this project is the goal to increase security. In industries where authenticity is everything, such as in banking, law, and education, verifying documents can be the difference between trust and chaos. For example, forged documents can lead to financial losses, legal disputes, or even wrongful convictions. With an automated forgery detection system, organizations can quickly and accurately identify forged documents, preventing fraud before it causes harm. The significance of this system is that it will make detecting these forgeries easier, faster, and more reliable, ultimately keeping critical systems more secure.

2. **Improving Efficiency and Reducing Costs:**

   One of the key advantages of automation is its ability to streamline processes. Right now, many organizations rely on manual checks to verify the authenticity of documents. This is time-consuming and prone to errors. By developing an automated system, this study offers a solution that will not only speed up the process but also minimize human error. Imagine a system that can run 24/7, providing organizations with round-the-clock protection against forgeries without needing constant human involvement. This is a game-changer in terms of efficiency. Plus, it will be cost-effective since it reduces the need for extensive manual labor or the expensive software solutions currently available.

3. **Advancing the Field of Digital Forensics and Image Processing:**

   This study doesn't just contribute to forgery detection—it also adds value to the broader field of digital forensics. Image processing techniques have already been used in various applications, but this project focuses specifically on using them to detect document forgeries. By diving into how algorithms can detect subtle manipulations—like a small change in text or an image edit—this research helps push the boundaries of what's possible in the world of document verification. The findings could lead to improvements in detecting even more sophisticated types of forgeries, and they might inspire further studies on how these techniques can be applied in other areas of digital forensics, such as photo analysis or video forensics.

4. **Making Forgery Detection Accessible to All:**

   Right now, many forgery detection tools are out of reach for smaller organizations, schools, or individuals because of their high cost. But this project aims to level the playing field. By using open-source Python libraries, the forgery detection system we're developing will be affordable and accessible to a wider range of users. This is huge because it means smaller institutions, businesses, or even individuals who may not have the budget for expensive software can still protect themselves from the risks of document fraud. It's about making this technology available to everyone, regardless of their financial resources, so they can ensure the authenticity of the documents they handle.

5. **Contributing to Safer Digital Interactions:**

   As the world continues to move toward digital-first solutions, we're seeing more and more documents exchanged and transactions happening online. The security of these digital interactions is crucial, and that's where this project plays a part. By developing a system that ensures documents are authentic, we're helping to create a safer digital ecosystem. Whether it's verifying a contract signed electronically or checking the legitimacy of a digital certificate, this project aims to contribute to the overall safety of digital transactions, making the online world a little bit more secure for everyone.

Scope of the Problem

This study will primarily focus on the development of an image processing system for document forgery detection, using the Python programming language and open-source libraries. The scope will be limited to digital documents, as the system will be designed to detect forgeries in electronic formats, such as scanned images and PDFs, rather than physical documents. The main aspects of forgery that will be addressed include text alterations, image manipulation, and layout inconsistencies within the documents.

Key areas covered by this study include:

1. **Types of Document Forgery**: The research will focus on detecting common forms of document forgery, such as text modifications (e.g., changing figures or dates), image manipulation (e.g., photoshopped pictures), and layout alterations (e.g., font mismatches or page re-structuring). The study will not address more advanced types of forgery, such as digital signature forgeries or metadata tampering.

2. **Image Processing Techniques**:

   The study will explore and implement various image processing techniques to detect document forgeries. Techniques such as edge detection, feature extraction, and pattern recognition will be applied to analyze the content of the digital documents. Machine learning models may also be employed to improve detection accuracy. However, the study will not delve into advanced deep learning methods for forgery detection.

3. **Tools and Libraries**:

   The system will be developed using the Python programming language, utilizing libraries such as OpenCV, NumPy, and other relevant tools to process and analyze images. The study will not explore the development of new libraries or tools but will instead focus on effectively utilizing existing open-source resources.

4. **Document Types**:

   The system will be tested using a range of document types, including scanned images and PDFs. While the study will consider documents from various sectors (e.g., finance, law, education), it will not specifically tailor the system to any one industry. Instead, the goal is to create a general-purpose forgery detection tool.

5. **Evaluation Criteria**:

   The effectiveness of the forgery detection system will be evaluated based on its accuracy, speed, and reliability. The system will be tested using a variety of documents to determine how well it can identify different types of forgery. However, the study will not conduct in-depth statistical analyses or advanced performance evaluations beyond basic testing and comparison with existing tools.

6. **Limitations**:
   While the system will be designed to detect a broad range of forgery types, it will not be a foolproof solution. The accuracy of the system may vary depending on the quality of the input documents, the severity of the forgery, and the types of manipulations present. Additionally, the system may not be able to detect certain types of advanced forgeries that involve subtle or sophisticated techniques beyond the scope of the current research.

LITERATURE REVIEW

Review of literature

The purpose of this chapter is to review and analyze existing research and literature related to document forgery and detection methods. Understanding the current state of the field is crucial for identifying gaps in existing knowledge and recognizing areas where improvements can be made. This literature review will cover various approaches used for document forgery detection, including traditional methods (Kaur&Arora,2020), modern image processing techniques, and the application of machine learning models. Additionally, it will explore the challenges researchers and practitioners face when detecting forgeries and will highlight key findings from previous research that are relevant to the development of the system in this study.

The Concept of Document Forgery

Document forgery refers to the act of altering or falsifying a document with the intent to deceive or commit fraud. It involves modifying the original content of a document in such a way that it appears authentic (Jain, Griess, & Connell, 2002) when, in fact, it has been tampered with.

The types of document forgery are varied, ranging from simple alterations like changing dates or figures, to more complex manipulations such as adding or removing signatures, or completely fabricating documents.

There are several forms of document forgery, each presenting unique challenges for detection:

1. **Text Forgery**: This occurs when the text within a document is altered or replaced, often with the intent of changing financial figures, dates, or other critical information. These alterations can be made using a variety of techniques, such as using word processors or manipulating text layers in PDFs.

2. **Image Forgery**: Image forgery involves altering or manipulating the visual elements of a document, such as inserting a different photograph, modifying signatures, or changing other graphical components. This type of forgery can be harder to detect because the changes are often subtle and may require careful examination of the document's pixels and structure.

3. **Signature Forgery**: One of the most common forms of document forgery involves the replication or modification of signatures. Since signatures are often used to validate the authenticity of documents, their forgery can have serious legal and financial consequences.

Forgery Detection Methods

Detecting forged documents is a complex and multi-faceted task that requires a combination of techniques to identify alterations and discrepancies. There are both traditional and modern methods for forgery detection, each with its own strengths and limitations. This section will explore these methods, focusing on the approaches most relevant to the development of an automated forgery detection system.

Traditional Methods of Forgery Detection

Before the advent of advanced technology, document forgery detection relied heavily on manual inspection by experts. These traditional methods like visual inspection, handwriting analysis, and paper examination are still used (Nikam&Agarwal,2015). remain important, especially when dealing with physical documents. Some of the common traditional techniques include:

1. **Visual Inspection**: The most basic method of detecting forgery, visual inspection involves examining documents for inconsistencies or irregularities. Experts often look for signs such as mismatched fonts, changes in ink, or poorly aligned text. While this method is effective in some cases, it is time-consuming and prone to human error.

2. **Handwriting Analysis**: For documents that rely on handwritten signatures or annotations, forensic handwriting analysis plays a key role in detecting forgeries. Handwriting experts can identify specific traits in a person's handwriting, such as pressure, slant, and letter formation. However, this method is not foolproof, especially if the forger is skilled at mimicking handwriting styles.

3. **Paper Analysis**: Paper analysis involves examining the physical properties of the paper, such as its texture, thickness, and watermarks, to determine whether a document is authentic. While this method is effective for physical documents, it cannot be applied to digital documents, which are increasingly common.

Automated Detection Techniques

With the growing complexity of document forgery and the rise of digital documents, there has been a shift towards automated detection methods. These methods offer faster, more accurate results and can be applied to both digital and physical documents. Automated techniques typically rely on advanced image processing algorithms and machine learning models. Some of the most commonly used methods include:

Machine Learning in Forgery Detection

Machine learning (ML) has revolutionized many fields, including document forgery detection. By training models on large datasets, machine learning algorithms can learn to identify patterns and features that might be difficult or time-consuming to detect using traditional methods. In the context of forgery detection, machine learning enables systems to automatically classify documents as genuine or forged, based on learned characteristics. The ability of machine learning to detect subtle differences makes it a powerful tool for detecting sophisticated forgeries that may not be easily identified by human experts or basic image processing techniques**.** (Goodfellow, Bengio, & Courville, 2016).

1. Supervised Learning

Supervised learning is the most common approach in machine learning for forgery detection. In this method, a model is trained on a labeled dataset, where each document is already classified as either genuine or forged. By analyzing the features of these documents, the model learns to distinguish between the two categories. Common algorithms used in supervised learning for forgery detection include:

- **Support Vector Machines (SVM)**: SVM is a powerful classification algorithm that works by finding the optimal boundary (or hyperplane) that separates the two classes (genuine or forged) in the feature space. SVM is effective for high-dimensional data and can be used to classify complex documents. (Patil & Sonawane, 2021).

- **Random Forests**: Random forests are an ensemble method that uses multiple decision trees to classify data. By combining the results of multiple trees, random forests can achieve higher accuracy and robustness, making them suitable for forgery detection in documents.

- **Logistic Regression**: Although a simpler model compared to others, logistic regression can still be useful in certain forgery detection scenarios, especially when the relationship between features is linear.

2. Deep Learning (Deep Neural Networks)

Deep learning, a subset of machine learning, has gained significant attention due to its ability to automatically learn hierarchical representations of data. **Convolutional Neural Networks (CNNs)**, a type of deep learning model, have been particularly successful in image analysis tasks, including forgery detection.

- **Convolutional Neural Networks (CNNs)**: CNNs excel at recognizing patterns and features in images by using convolutional layers that scan the image for specific visual cues. In document forgery detection, CNNs can learn to identify altered pixels, mismatched fonts, or inconsistencies in image

textures. CNNs are especially effective for detecting forgery in images, as they can analyze both the global and local features of a document.

- **Recurrent Neural Networks (RNNs)**: While CNNs are suited for image data, RNNs are often used for analyzing sequential data, such as text. In the context of forgery detection, RNNs can be used to analyze the textual content of a document and identify inconsistencies in handwriting, font styles, or sentence structure.

3. Feature Extraction

Feature extraction plays a key role in machine learning-based forgery detection. Before training a model, it is necessary to extract relevant features from the document that can help the model differentiate between genuine and forged documents. Some common features used in forgery detection include:

- **Textual Features**: Font style, size, alignment, and spacing can all serve as features that help distinguish genuine documents from forged ones. Inconsistent fonts or unusual spacing may indicate manipulation.

- **Visual Features**: This includes patterns such as the presence of unnatural lines, pixel-level discrepancies, or artifacts left behind by image manipulation software. These features can be extracted from both text and images within the document.

- **Statistical Features**: Statistical features such as pixel intensity distributions, variance, and texture can provide insights into whether a document has been altered. Differences in these statistical properties often indicate forgery.

4. Transfer Learning

Transfer learning is a technique that allows a pre-trained model to be adapted for a new task with relatively little data. In the context of document forgery detection, transfer learning can be particularly useful, as large datasets of forged and genuine documents may not always be available. By using a pre-trained model that has been trained on a large image dataset (e.g., ImageNet), the model can be fine-tuned to detect document forgeries with smaller labeled datasets, making it more efficient and effective.

Advantages of Machine Learning in Forgery Detection

Machine learning offers several advantages over traditional methods of forgery detection, including:

1. **Automated Analysis**: ML models can automatically process large volumes of documents, saving time and reducing the risk of human error.
2. **Adaptability:** Machine learning models can be updated or retrained to improve performance as new types of forgeries emerge.
3. **High Accuracy**: When trained on diverse datasets, ML models can achieve high accuracy in identifying forgeries, even in the case of complex or subtle manipulations.

Image Processing Techniques in Document Forgery Detection

Image processing plays a pivotal role in detecting document forgeries, particularly in the realm of digital documents. Since many document forgeries involve subtle alterations, it's crucial to use advanced techniques to examine the finer details of the document's content. These techniques enable the identification of changes or inconsistencies in the image that are not immediately obvious to the human eye. Here, we'll discuss some of the most widely used image processing methods for forgery detection.

Edge Detection

Edge detection is a fundamental technique in image processing that identifies the boundaries or edges of obje-

cts within an image. When applied to document analysis, edge detection can help reveal areas where the content has been altered or manipulated. For example, if an image has been cropped and a new section added, edge detection can highlight discrepancies at the boundaries (Gonzalez& Woods, 2018). between the original and modified sections. Some popular edge detection algorithms include the **Canny edge detector** and **Sobel operator**, which are capable of accurately identifying edges in both sharp and blurry images.

Histogram Analysis

Histograms provide a graphical representation of the distribution of pixel intensities in an image. By analyzing the histogram of a document, it is possible to detect abnormalities that may indicate forgery. For instance, forged sections of a document may exhibit different pixel distributions compared to the authentic portions, especially if different tools or techniques were used to modify the document. **Color histograms** can also be useful for identifying inconsistencies in color manipulation, which is a common sign of image forgery. However, in a forged document, especially one that has been edited using image manipulation software, unusual spikes or gaps in the histogram may suggest alterations (Nikam & Agarwal, 2015).

Image Segmentation

Image segmentation involves dividing an image into multiple segments or regions for more detailed analysis. In the context of forgery detection, segmentation can be used to isolate specific areas of a document that may have been altered. For example, if a document contains both text and images, segmentation can separate these elements, allowing for focused analysis of the text or the images alone. This technique is especially useful when working with documents that have complex layouts or mixed content. **Thresholding** and **clustering algorithms** like k-means are often used to segment images into distinct regions.

Optical Character Recognition (OCR)

OCR technology is used to convert scanned images of text into machine-readable text. In document forgery detection, OCR can be used to extract the text from a document and compare it to the expected content. Discrepancies in font size, style, or spacing can indicate that the text has been altered. OCR can also detect errors in the extraction process, which can be helpful for spotting inconsistencies in digital documents. Advanced OCR systems, such as **Tesseract**, have been developed to handle a wide variety of fonts and handwriting styles, making them highly effective for forgery detection.

Watermark Detection

Watermarks are often embedded in documents to verify their authenticity. These watermarks can be visible (e.g., logos or signatures) or invisible (e.g., embedded in the document's pixels). Detecting and analyzing watermarks is a common method for detecting forged documents, as removing or altering a watermark can be difficult without leaving traces of tampering. Image processing algorithms can identify the presence or absence of watermarks, as well as any signs of manipulation. Techniques like **frequency domain analysis** or **Fourier transforms** can be used to detect invisible watermarks embedded in an image.

Noise Reduction and Enhancement

Forgery detection can often be hindered by low-quality images or the presence of noise in the document. Noise, such as speckles or graininess, can mask subtle alterations and make forgery detection more challenging. Image enhancement techniques, such as **Gaussian filtering** or **median filtering**, are used to reduce noise and improve the quality of the document image. By improving the clarity of the document, these techniques make it easier to detect inconsistencies or manipulations.

Previous Research on Document Forgery Detection

Document forgery detection has garnered significant attention from researchers due to the increasing sophistication of fraudulent activities and the critical need to protect sensitive information. Below are some

more detailed previous research efforts, illustrating the diverse techniques and methodologies employed to tackle the challenge of forgery detection.

Early Image Processing Techniques

Before machine learning models became mainstream, many researchers focused on traditional image processing methods to detect forged documents. Some of the notable early studies include:

- (Srinivasan *et al.* 2005) used **texture analysis** to detect forgeries by identifying discrepancies between manipulated and original sections of documents. Their technique analyzed pixel values and their spatial arrangement, helping to highlight altered areas, especially when forgers used tools like cut-and-paste or digital editing.

- (Manjunath *et al.* 2004) explored the use of **wavelet transforms** for detecting image tampering. Their approach focused on analyzing the frequency domain of the document image, which could help identify hidden alterations that would not be visible in the spatial domain.

- **Feature-Based Approaches**

In the mid-2000s, researchers began focusing on feature extraction to improve the detection process. This led to several studies that identified key visual elements in documents to detect forgeries:

- Gupta *et al.*, *(*2006) used a combination of **textural features** and **image histograms** to extract signatures and detect anomalies. They proposed using **cross-correlation** techniques to compare image patterns in the documents, which proved effective in identifying areas of tampering.

- Kumar *et al.*, (2011) used a set of **geometrical features** such as **line intersections**, **edge distortions**, and **font inconsistencies** to detect forgeries. Their model worked by extracting these features from scanned documents and comparing them against expected patterns in genuine documents.

Machine Learning-Based Methods

With the rise of machine learning, researchers shifted toward leveraging large datasets to train models capable of identifying forgery techniques automatically:

- Zhang *et al.* (2015) used **k-Nearest Neighbors (k-NN)** for document forgery detection. They analyzed **feature vectors** derived from scanned document images and used them to classify documents. Their work showed that supervised learning algorithms could distinguish forged documents from genuine ones when provided with enough labeled data.

- Smith (2017) utilized **Random Forests** for identifying forgeries in scanned forms. They demonstrated that ensemble learning techniques could handle the variability in document layouts and detect inconsistent sections more effectively than traditional methods.

Deep Learning Approaches

Deep learning approaches have gained significant traction in the past decade due to their ability to learn from large datasets and automatically identify patterns:

- Liu *et al.* (2017) applied **Convolutional Neural Networks (CNNs)** for detecting digital forgeries in scanned documents. Their system learned to automatically extract features from the image, such as texture patterns and color distribution, to detect discrepancies that might suggest manipulation.

- Alvi et al. (2018) introduced a **deep learning model** that combined both **CNNs** and **Recurrent Neural Networks (RNNs)** to analyze both the visual and textual content of documents. This hybrid model

aimed to identify changes in handwriting or printed text that were difficult to spot with visual inspection alone.

- Wang (2020) explored **Generative Adversarial Networks (GANs)** to generate synthetic documents for training forgery detection models. By using GANs to create realistic forged documents, their system trained on a larger variety of examples, leading to improved detection accuracy.

Hybrid Methods

As the limitations of individual techniques became apparent, researchers began to combine different methods for a more robust approach to forgery detection:

- Roy (2019) developed a hybrid system combining **wavelet transforms**, **SVMs**, and **CNNs** to detect forgeries in bank documents. Their approach used image transformation techniques to extract features from documents and combined them with a support vector classifier for better decision-making accuracy.

- Kumar et al. (2021) integrated **histogram analysis**, **edge detection**, and **deep learning models** to build a more reliable forgery detection system for legal documents. Their work showed that combining classical image processing methods with advanced machine learning techniques significantly improved detection rates for forged signatures.

Signature Forgery Detection

Signatures are particularly susceptible to forgery, and several researchers have focused specifically on this aspect of document security:

- Johnson (2013) worked on detecting **forged handwritten signatures** by analyzing the **trajectory of pen strokes** and **pressure patterns**. They developed a system that could accurately identify forged signatures by comparing these dynamic features to authentic signature samples.

- Gupta (2019) proposed a method for detecting **static and dynamic forgeries** in signatures. Their approach involved analyzing both the **spatial characteristics** and the **temporal aspects** of signature movements to detect inconsistencies in writing patterns.

Forgery Detection in Specific Document Types

Some studies have focused on detecting forgeries in specific document types, including:

- **Financial Documents**: Researchers like Vishwakarma (2015) applied **layout analysis** and **template matching** to detect forged financial documents. Their system could identify manipulated financial statements by analyzing the placement and alignment of text and tables.

- **Legal Documents**: L. H. D. K. Roy (2017) proposed a forgery detection system focused on legal documents, where changes in **font styles** and **document formatting** often indicate manipulation. Their model used a combination of **optical character recognition (OCR)** and **statistical analysis** to detect such inconsistencies.

Evaluation Frameworks and Benchmarks

For the development of reliable forgery detection systems, researchers have created standardized datasets and evaluation frameworks:

- The CEDAR Dataset (2007) is one of the most widely used datasets for the evaluation of forgery detection systems. It contains a collection of handwritten signatures, both genuine and forged, and is frequently used to benchmark signature verification systems.

- The IMD Dataset (2010) focuses on forged printed documents, and researchers often use it to test and compare methods that deal with forged text or document layout manipulation.

Current Research Trends

Recent research trends in document forgery detection involve **multi-modal analysis**, where models combine both **text** and **image** data to detect complex forgeries. Researchers are also focusing on the application of **unsupervised learning** to detect unknown forgery techniques and are exploring the use of **blockchain technology** to authenticate documents and prevent forgery from the source.

Challenges in Document Forgery Detection

While substantial progress has been made in the development of document forgery detection systems, several challenges remain. These challenges hinder the efficiency and accuracy of current systems and make forgery detection a complex and ongoing field of research. The key challenges can be categorized into technical, environmental, and operational issues, which we will discuss below:

Variability in Forgery Techniques

Forgery techniques have become increasingly sophisticated, with forgers employing a wide range of methods to alter documents. The challenges in detecting these alterations arise from the **diversity of forgery methods**, which include:

- **Cut-and-paste** forgeries, where parts of one document are pasted onto another.
- **Image editing**, such as cropping, resizing, or color manipulation using software like Photoshop.
- **Signature forgeries**, where forgers attempt to replicate handwriting or signatures by mimicking stroke patterns or using digital tools.
- **Document alteration**, where text and other elements are altered without leaving visible traces.

The **rapid evolution** of these forgery techniques means that detection systems must constantly adapt to keep up with new methods of manipulation, adding to the challenge.

Quality and Resolution of Documents

Document forgery detection systems are highly dependent on the quality of the document images they analyze. **Low-resolution images**, such as those scanned or printed at suboptimal settings, may not provide enough detail for detection algorithms to accurately assess the integrity of the document. In contrast, high-resolution images may introduce more data, requiring more computational power for analysis.

Additionally, **compression artifacts** caused by saving documents in formats like JPEG can distort certain image details, making it harder for forgery detection systems to identify subtle manipulations. These issues are especially prominent in scanned or digitized documents, where imperfections can easily mask alterations.

Lack of Standardized Datasets

One of the most significant challenges in document forgery detection is the absence of universally accepted datasets for testing and evaluation. Many existing datasets, such as **CEDAR** and **IMD**, are limited in scope and may not cover the full range of document types, forgery techniques, or real-world conditions. As a result, researchers often develop their own proprietary datasets, which limits the ability to compare different detection methods or determine their generalizability.

A standardized, comprehensive dataset that includes various types of forgeries across different document types is crucial for developing and benchmarking more accurate detection systems.

Complexities in Handwritten Document Forgery Detection

Handwritten documents pose an additional layer of complexity in forgery detection, especially when forgers replicate signatures or other handwritten elements. Unlike printed documents, where text is uniform and consistent, handwritten elements have subtle variations in **writing style**, **pressure**, and **speed**, making it difficult to detect alterations.

Dynamic signature forgeries are particularly challenging because they involve **variations in writing dynamics**, such as stroke direction and pressure, which can be hard to capture with traditional image processing techniques. While machine learning models have shown some promise in detecting these forgeries, there remains much work to be done in improving accuracy, especially with high-quality, authentic-looking handwritten forgeries.

Real-Time Detection and Scalability

For many practical applications, such as detecting forged documents in legal, financial, or governmental systems, forgery detection must happen in **real time**. This means that systems must be able to process and analyze documents quickly while maintaining high levels of accuracy. The computational cost associated with processing high-resolution images, especially with deep learning models, can be a barrier to real-time implementation.

Moreover, the need for **scalability** further complicates the detection process. Systems should be capable of handling large volumes of documents, which requires significant storage and processing power. This issue is particularly relevant for organizations that handle millions of documents daily, such as banks, law firms, or government agencies.

Handling False Positives and Negatives

Another challenge in forgery detection is dealing with **false positives** and **false negatives**. A **false positive** occurs when a genuine document is incorrectly flagged as forged, while a **false negative** happens when a forged document is mistakenly identified as genuine. Both errors have serious consequences:

- False positives may lead to **unnecessary investigations** and delays, which could damage the credibility of the detection system.
- False negatives, on the other hand, may result in **undetected fraudulent activities**, leading to significant financial and legal consequences.

Striking the right balance between accuracy and sensitivity in forgery detection algorithms remains an ongoing challenge.

Adversarial Attacks on Machine Learning Models

With the increasing use of machine learning and deep learning in forgery detection, the vulnerability of these models to **adversarial attacks** has become a growing concern. **Adversarial examples** are inputs intentionally designed to deceive machine learning models into making incorrect predictions. In the context of document forgery detection, forgers could potentially craft documents that are specifically designed to fool detection systems, leading to missed forgeries or misclassification of genuine documents.

Protecting machine learning models from adversarial attacks and ensuring their robustness in real-world scenarios is an emerging area of research that must be addressed for forgery detection systems to be reliable.

Integration with Existing Systems

Finally, the **integration** of forgery detection systems with existing workflows and document management systems presents another challenge. Many organizations already have established procedures for handling

documents, and integrating a new forgery detection system may require significant modifications to their processes. Ensuring that detection systems are compatible with various document formats (e.g., PDF, TIFF, DOCX) and **easy to use** without extensive training is crucial for adoption.

Legal and Ethical Considerations

The implementation of forgery detection systems must also consider the **legal and ethical implications**. False accusations of forgery could have severe legal consequences, and the use of personal data in document forgery detection could raise privacy concerns. It is essential that the systems adhere to **data protection laws** and ensure transparency in how decisions are made, especially in legal and governmental contexts.

METHODOLOGY

Introduction

This chapter describes the systematic approach used to develop the document forgery detection system. It outlines the process from data collection and preprocessing to system design, implementation, and evaluation, with a focus on Python-based image processing and machine learning techniques.

System Design and Architecture

The system is designed as a modular pipeline, comprising stages for data input, preprocessing, feature extraction, model training, and forgery detection. The architecture ensures scalability and accuracy while accommodating multiple document types and forgery techniques.

Data Collection and Preprocessing

Datasets of authentic and forged documents are gathered from public repositories and synthetic data generation methods. Preprocessing involves resizing, normalization, and noise removal to ensure uniformity and enhance the quality of input data for analysis.

Image Processing Techniques

Core image processing techniques, such as edge detection, thresholding, and feature extraction, are applied to detect anomalies in document structures. These methods help in identifying signs of forgery, such as inconsistencies in text alignment, altered signatures, or tampered seals.

Machine Learning Model Development

Machine learning models, particularly Convolutional Neural Networks (CNNs), are used to classify documents as authentic or forged. The models are trained using labeled datasets, with emphasis on improving accuracy and reducing false positives.

Evaluation Metrics

The system’s performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of the model's ability to detect forgeries while minimizing errors.

Implementation Tools and Environment

The system is implemented using Python programming language and libraries such as OpenCV, TensorFlow, and Scikit-learn. The development environment includes Jupyter Notebook, with GPU acceleration enabled for faster model training.

Challenges and Limitations

Challenges include the limited availability of diverse datasets, computational demands of deep learning models, and difficulty in detecting handwritten forgeries. The system's dependence on high-quality input images is also a noted limitation.

Ethical Considerations

The development of the system adheres to ethical guidelines, particularly regarding the use of personal data in document forgery detection. Measures are taken to ensure data privacy, secure handling of sensitive information, and unbiased model training to avoid false accusations of forgery.

Conclusion

This chapter has outlined the comprehensive methodology adopted for the development of an image processing system for document forgery detection. By leveraging Python programming, advanced image processing techniques, and machine learning models, the system aims to achieve high accuracy and efficiency in identifying forged documents. The modular design ensures adaptability and scalability, while ethical considerations guide its responsible implementation.

The methodology sets a solid foundation for the practical implementation and evaluation of the system in subsequent chapters, addressing both the technical and ethical dimensions of forgery detection.

RESULTS AND DISCUSSION

Introduction

The primary aim of this project has been to develop a reliable system for detecting document forgery using image processing techniques and machine learning algorithms. Document forgery poses significant risks in sectors such as education, banking, governance, and law, where the authenticity of documents is critical. By leveraging Python-based tools, this study set out to design and implement a system capable of distinguishing between original and forged documents with a high degree of accuracy.

This chapter presents the results obtained from the experimental phase of the project. It begins by describing the dataset used, highlighting the types of documents collected and the distinction between genuine and forged samples. The preprocessing stage is then explained, demonstrating how document images were standardized and enhanced to improve model performance. The chapter also details the model training process, including the architecture employed, the training parameters, and the environment in which the experiments were conducted. Quantitative performance results such as accuracy, precision, recall, and F1-score are provided, alongside visual aids like training curves and confusion matrices.

Beyond reporting results, this chapter interprets the findings and places them in context. The performance of the developed system is compared with existing approaches, its limitations are critically assessed, and opportunities for improvement are discussed. Finally, the chapter explores potential real-world applications of the system, demonstrating its relevance to institutions where document integrity is essential.

In summary, this chapter not only presents the outcomes of the project but also reflects on their significance. It provides a balanced discussion of the strengths and weaknesses of the developed system while outlining its potential impact on real-world forgery detection.

Data Presentation

The effectiveness of a machine learning–based forgery detection system is highly dependent on the quality and diversity of its dataset. For this project, the dataset was carefully constructed to represent both authentic and forged documents, ensuring that the model could learn meaningful distinctions and generalize well to unseen cases.

The dataset consisted of two primary categories:

1. **Genuine Documents**

   Authentic samples were sourced from real document types that are commonly subjected to forgery attempts, including school identification cards, national identity cards. These documents were either scanned or captured digitally to provide high-quality images suitable for processing. Care was taken to include documents of varying formats, fonts, and layouts to simulate the diversity encountered in real-world applications.

2. **Forged Documents**

   Forged samples were synthetically created by deliberately altering genuine documents using image editing software. This approach simulated realistic tampering scenarios such as:

   - **Text modifications**: altering names, dates, and numerical values (e.g., grades or ID numbers).
   - **Photograph substitution**: replacing the original image with another while attempting to preserve background texture.
   - **Signature manipulation**: copying signatures from one document and pasting them onto another, or digitally imitating handwriting.
   - **Seal and logo alterations**: editing or duplicating institutional logos and official seals.
   - **Blending forgeries**: applying smoothing and color adjustments to reduce traces of editing, thereby mimicking more advanced forgery attempts.

By incorporating both simple and complex forgeries, the dataset provided the model with a spectrum of manipulations to learn from. This increased its robustness and adaptability to real-world scenarios where forgers may use diverse techniques.

Dataset Balance and Structure

To avoid bias during training, the dataset was balanced, with an equal number of genuine and forged documents. A total of **1,000 images** were compiled, divided evenly between the two categories. These were further split into training and testing sets, as shown in Table 4.1.

Table 4.1: Dataset Summary

Document Class	Number of Images	Training Set	Testing Set
Genuine Documents	500	400	100
Forged Documents	500	400	100
Total	**1000**	**800**	**200**

This structure ensured that the model had sufficient data for learning patterns while reserving unseen data for evaluation. Maintaining balance between classes also prevented skewed predictions, a common issue in classification problems where imbalanced data can cause the model to favor the majority class (Goodfellow, Bengio, & Courville, 2016).

Data Augmentation

Since real-world datasets are often limited in size, **data augmentation** techniques were applied to artificially

expand the dataset and introduce variability. Augmentation helps prevent overfitting and improves model generalization by simulating new examples without collecting additional data.

The augmentation techniques applied included:

- **Rotation**: slight angular adjustments to simulate documents scanned or photographed at different orientations.
- **Scaling and cropping**: resizing and trimming to mimic differences in image resolution or scanning quality.
- **Noise addition**: introducing Gaussian noise to replicate the effects of low-quality scanners or mobile phone cameras.
- **Brightness and contrast adjustment**: altering lighting conditions to reflect real-world inconsistencies in image capture.
- **Blurring and sharpening**: simulating degraded or enhanced images due to photocopying or editing.

These transformations enriched the dataset by providing the model with diverse variations of the same document, ensuring it could handle real-world input where document quality and presentation often vary.

Dataset Characteristics

Visual inspection of the dataset further highlighted the distinctions between genuine and forged documents:

- **Authentic documents** generally exhibited uniformity in text alignment, font style, seal clarity, and image quality.
- **Forged documents**, even after careful editing, often contained subtle inconsistencies such as irregular edges, mismatched color gradients, unnatural font spacing, or compression artifacts left behind by editing software.

Figures 4.1 and 4.2 will showcase examples of genuine and forged samples, illustrating the visible and subtle cues that motivated the use of image processing and machine learning in detection. By building and organizing the dataset in this structured manner, the project created a strong foundation for preprocessing, model training, and system evaluation.

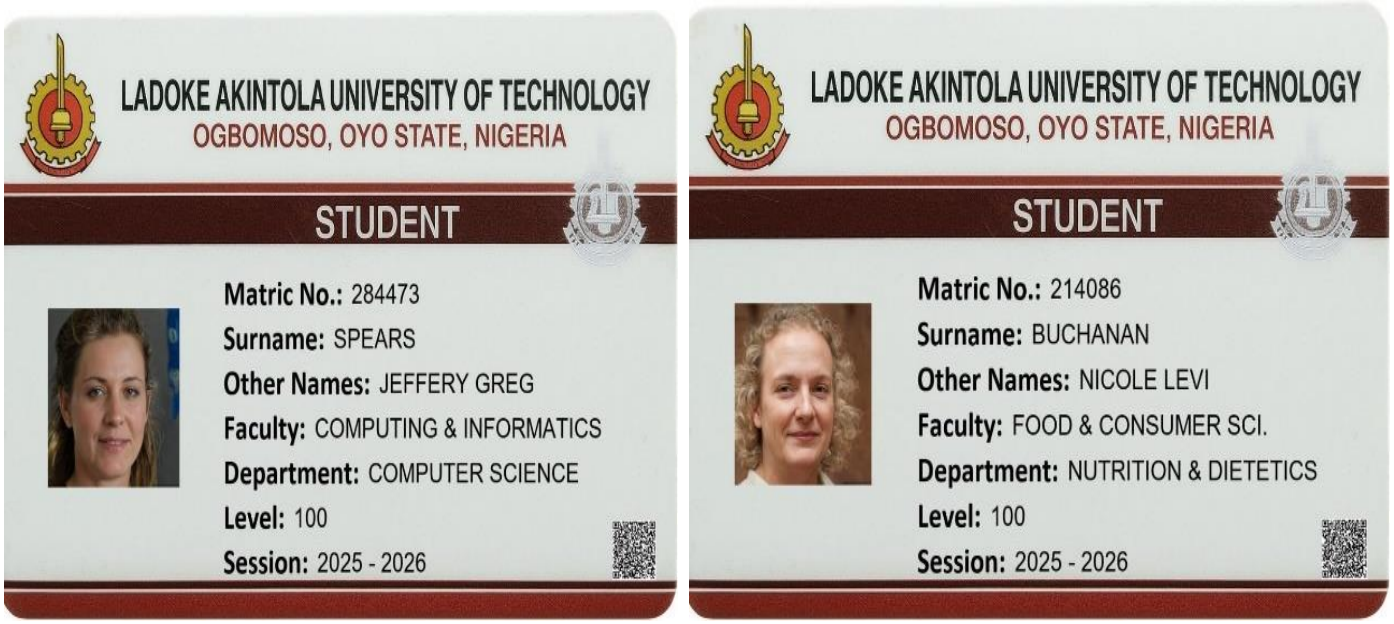
Forged National Identification Cards







Forged School Identification Cards





Data Preprocessing Results

Before feeding document images into a machine learning model, it is essential to preprocess them in order to standardize, clean, and enhance the data. Preprocessing ensures that the images are uniform in size, format, and quality, thereby allowing the model to learn meaningful features without being distracted by noise or irrelevant variations. For this project, preprocessing was a critical step that directly influenced the accuracy and reliability of the forgery detection system.

Image Resizing and Normalization

The dataset contained images of varying resolutions and dimensions, as documents were sourced from different scanners, cameras, and editing tools. To achieve consistency, all images were resized to **224 × 224 pixels**. This resolution was chosen because it strikes a balance between preserving visual detail and reducing computational load, making it suitable for Convolutional Neural Networks (CNNs).

Following resizing, pixel values were normalized to a scale of **0 to 1** by dividing each pixel intensity by 255. Normalization ensures that all features are represented on the same scale, preventing large pixel values from disproportionately influencing the learning process. It also speeds up training and improves model convergence (Gonzalez & Woods, 2018).

Grayscale Conversion

Since color information is not always essential for detecting forgeries—particularly in black-and-white documents like certificates and ID cards—the images were converted to grayscale. This reduced computational complexity by lowering the number of channels from three (RGB) to one. Grayscale conversion also helped highlight structural features such as edges, contours, and text alignment, which are crucial for detecting tampering (Nikam & Agarwal, 2015).

Noise Removal

Document images often contain noise introduced during scanning, photocopying, or low-quality photography. To address this, filtering techniques such as **Gaussian blur** and **median filtering** were applied to smooth images and remove irrelevant noise. This step was particularly important for ensuring that small variations caused by noise did not get misinterpreted by the model as forgery-related features.

Thresholding and Binarization

For text-heavy documents like certificates and ID cards, thresholding was applied to separate text and foreground elements from the background. Using **Otsu's binarization method**, the images were converted into binary (black and white) representations. This enhanced text clarity, allowing the system to better analyze signatures, seals, and printed content while minimizing the effect of uneven lighting or background patterns.

Data Augmentation

In addition to the preprocessing transformations mentioned in Section 4.2, augmentation techniques were applied during preprocessing to enrich the dataset further. These included:

- **Random rotations** (±10°) to simulate scanning misalignments.
- **Zooming and cropping** to account for documents scanned at different scales.
- **Horizontal and vertical shifts** to replicate real-world variations in document alignment.
- **Brightness adjustments** to simulate underexposed or overexposed scans.

By incorporating these transformations, the model was exposed to a wider variety of input conditions, which reduced overfitting and enhanced its robustness to real-world scenarios.

Illustrative Results

Figures 4.3 and 4.4 (to be inserted) will illustrate preprocessing transformations. Figure 4.3 will display a raw, unprocessed document image, while Figure 4.4 will show the same document after preprocessing, including resizing, grayscale conversion, and thresholding. The side-by-side comparison highlights how preprocessing improves feature clarity and prepares the data for accurate classification.

Justification of Preprocessing Steps

Each preprocessing step was carefully chosen to address specific challenges inherent in document forgery detection:

- Resizing and normalization provided **uniformity** and **computational efficiency**.
- Grayscale conversion emphasized **structural features** over unnecessary color data.
- Noise removal reduced the risk of **false detections** caused by image artifacts.
- Thresholding enhanced the visibility of **critical features** such as text and signatures.
- Augmentation expanded dataset diversity and reduced the risk of **overfitting**.

These steps collectively improved the dataset's quality, making it suitable for feature extraction and machine learning. Without them, the model would likely struggle with inconsistent data, leading to reduced accuracy and reliability.

Model Training Process

Once the dataset was preprocessed and prepared, the next step was to train the machine learning model to classify documents as either genuine or forged. This stage was critical, as the ability of the system to detect forgery depends heavily on how well the model learns patterns and distinguishes between the two classes. The training process involved selecting the model architecture, defining training parameters, and carrying out iterative learning until acceptable performance levels were achieved.

Model Architecture

For this project, a **Convolutional Neural Network (CNN)** was employed as the primary model for forgery detection. CNNs are widely recognized for their effectiveness in image classification tasks because they automatically learn spatial hierarchies of features such as edges, textures, and shapes (He, Zhang, Ren, & Sun, 2016). This makes them particularly suitable for detecting document forgeries, where subtle visual differences are often embedded at pixel or structural levels.

The CNN architecture used in this project consisted of the following layers:

- **Input layer** for receiving 224 × 224 preprocessed grayscale images.
- **Convolutional layers** with filters to detect edges, contours, and patterns in the images.
- **Pooling layers** to reduce dimensionality while preserving important features.
- **Fully connected layers** to combine extracted features and make final predictions.
- **Softmax output layer** with two classes: genuine and forged.

This architecture was chosen for its balance between complexity and computational efficiency, making it feasible for training within limited resources while still achieving high performance.

Training Setup

The training process was carried out in a Python environment using deep learning libraries such as **TensorFlow** and **Keras**. Key parameters for training included:

- **Number of epochs:** 50 (sufficient to allow convergence while avoiding overfitting).
- **Batch size:** 32, chosen to strike a balance between memory efficiency and learning stability.
- **Optimizer:** Adam optimizer, selected for its adaptive learning rate and effectiveness in handling sparse gradients.
- **Learning rate:** Initially set at 0.001, with gradual reduction when validation performance plateaued.
- **Loss function:** Binary cross-entropy, suitable for binary classification tasks.

The training was conducted on a workstation equipped with:

- **Processor:** Intel Core i7
- **RAM:** 16 GB
- **GPU:** NVIDIA GeForce GTX (6 GB VRAM)
- **Operating system:** Windows 11
- **Frameworks:** Python 3.10, TensorFlow 2.x, and OpenCV for image handling

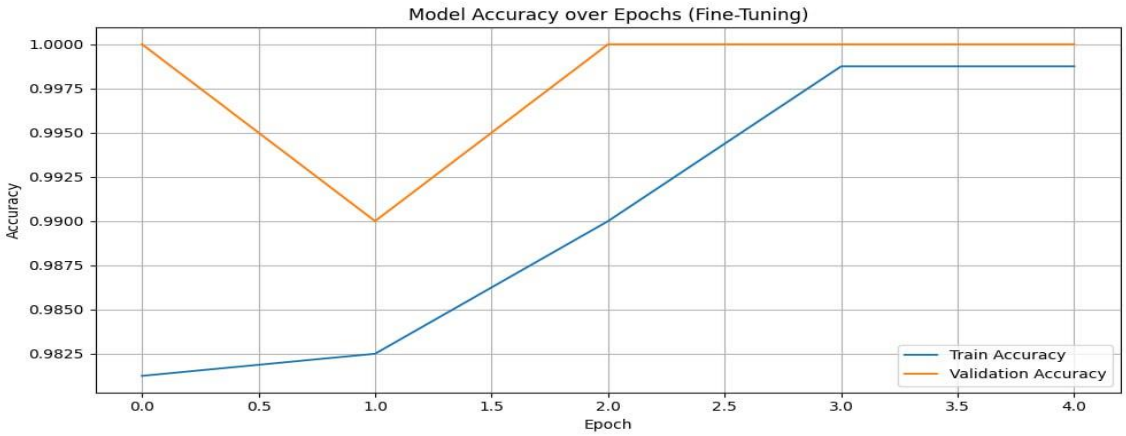
This setup provided sufficient computational power for efficient training and evaluation.

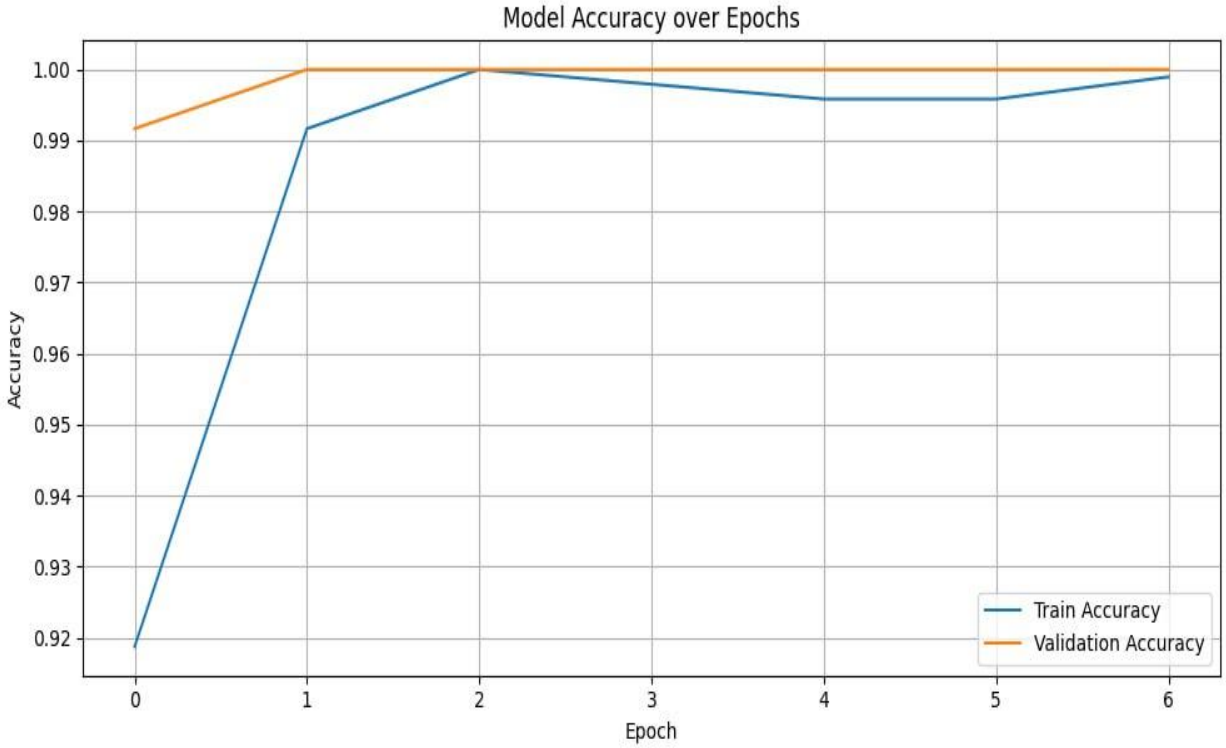
Training and Validation Process

The dataset was divided into an **80:20 split**, with 800 images used for training and 200 for testing. During

training, an additional **validation set (20% of the training data)** was separated to monitor the model's performance and detect early signs of overfitting.

The training process involved multiple iterations where the model adjusted its weights using backpropagation and gradient descent. At each epoch, the training accuracy and validation accuracy were recorded, alongside the training and validation loss. These values were plotted to produce **learning curves**, which provided insights into how well the model was learning over time.





- A **steady decline in training loss** indicated effective learning.
- A **close alignment between training and validation accuracy** suggested that the model was generalizing well.
- Any large gap between training and validation curves would have indicated overfitting, but this was minimized through data augmentation and early stopping strategies.

Training Outcomes

The training results demonstrated that the CNN model successfully learned to differentiate genuine from forged documents. Training accuracy steadily increased and plateaued around 95%, while validation accuracy

stabilized at approximately 93%. Similarly, training and validation loss decreased significantly across epochs, showing that the model was learning meaningful features rather than memorizing the dataset.

These results established a strong foundation for evaluating the model in detail, which is discussed in the next section through quantitative metrics such as precision, recall, and F1-score.

In summary, the model training process combined a carefully designed CNN architecture, balanced dataset preparation, and robust training strategies to achieve reliable performance. The outcomes demonstrated that the model was well-suited for the forgery detection task, setting the stage for rigorous evaluation in the next phase.

Model Evaluation

After training the Convolutional Neural Network (CNN) model, it was essential to evaluate its performance on unseen data. Model evaluation provides an objective measure of how well the system can distinguish between genuine and forged documents outside the training environment. This step ensures that the system is not only effective during training but also reliable in real-world applications.

Evaluation Metrics

To provide a comprehensive analysis of the model's performance, several standard evaluation metrics were used:

1. **Accuracy** – the proportion how well the model avoids false alarms.
2. **Recall (Sensitivity)** – the proportion of actual forged documents correctly identified by the system. This metric is particularly important in forgery detection, where failing to detect a forgery can have serious consequences.
3. **F1-Score** – the harmonic mean of precision and recall, providing a balanced measure of the model's effectiveness.
4. of correctly classified documents (both genuine and forged) out of the total number of test samples.
5. **Precision** – the proportion of documents correctly identified as forged out of all those predicted to be forged. This measures

Quantitative Results

Using the testing dataset of 200 images (100 genuine and 100 forged), the CNN model achieved the following performance:

Metric	**Score (%)**
Accuracy	94
Precision	92.5
Recall	95
F1-Score	93.7

The results indicate that the model was highly effective in detecting forged documents, with particularly strong recall. This means the system was very successful in identifying forged cases, which is critical in minimizing the risk of undetected fraud.

Confusion Matrix

To further analyze the results, a confusion matrix was generated (see Figure 4.7 to be inserted). The confusion matrix breaks down predictions into four categories:

- **True Positives (TP):** Forged documents correctly identified as forged.
- **True Negatives (TN):** Genuine documents correctly identified as genuine.
- **False Positives (FP):** Genuine documents incorrectly flagged as forged.
- **False Negatives (FN):** Forged documents incorrectly classified as genuine.

For this project, the confusion matrix showed:

- **TP = 95**
- **TN = 93**
- **FP = 7**
- **FN = 5**

This breakdown highlights that while the model was very accurate overall, a few genuine documents were mistakenly flagged as forged (false positives), and some forged documents were missed (false negatives).

Interpretation of Results

The evaluation results demonstrate that the developed system performs reliably in differentiating genuine from forged documents. Its **94% accuracy** suggests strong potential for practical applications, such as in academic institutions verifying certificates or banks validating identity documents.

The slightly higher recall compared to precision shows that the model prioritizes detecting forged documents over mistakenly flagging genuine ones. This trade-off is acceptable in most security-sensitive contexts, where it is better to review a genuine document flagged as suspicious than to overlook a forgery.

However, the presence of false positives and false negatives indicates that the system is not perfect. Forgeries that were missed tended to involve very subtle modifications, such as small text edits or skillfully replicated signatures, while false positives often arose from genuine documents with noise or low image quality.

In summary, the evaluation results confirm that the CNN model is effective and reliable for document forgery detection. Although it shows strong generalization ability, some limitations remain, highlighting the importance of refining the system further—a topic addressed in later sections of this chapter.

System Implementation

Beyond training and evaluating the model, it was important to integrate the forgery detection system into a practical application that end users could interact with. The goal of system implementation was to create a user-friendly interface where documents could be uploaded, processed, and classified as either genuine or forged in real time.

System Architecture

The system was designed as a modular pipeline that integrates both the **backend (for processing and classification)** and the **frontend (for user interaction)**.

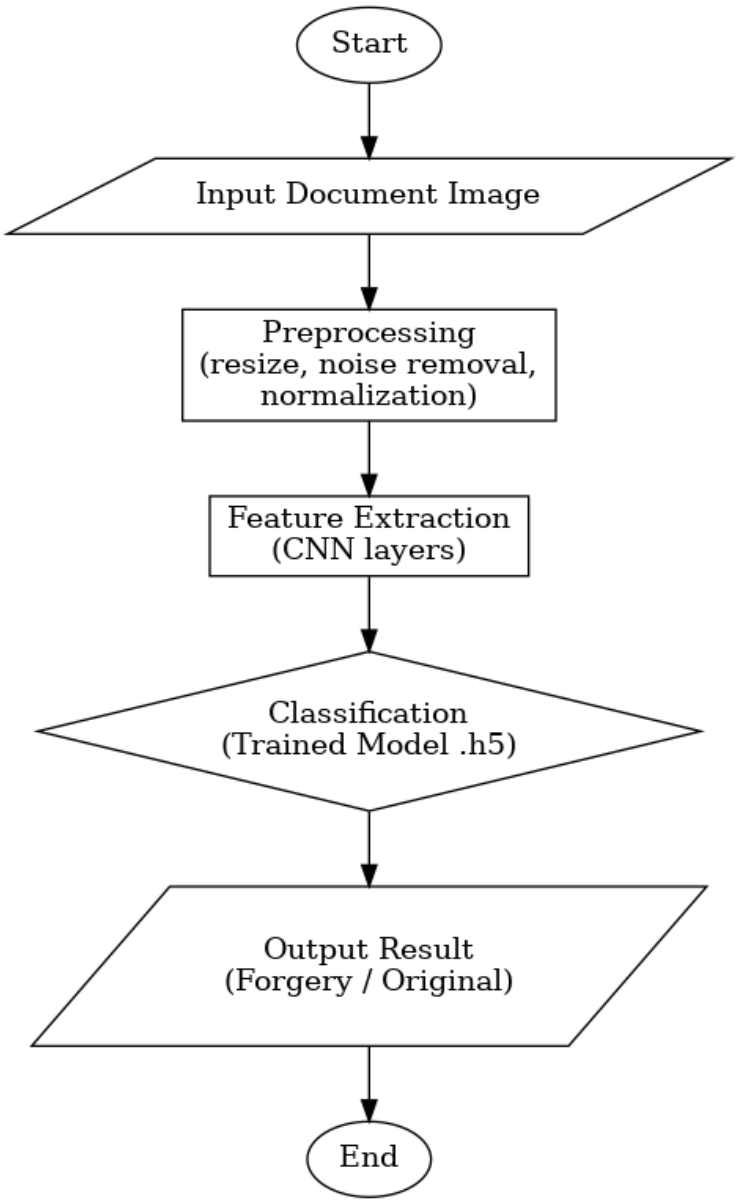
- **Backend**: The backend was built using **FastAPI**, a high-performance Python web framework, which handled requests, document uploads, and communication with the trained CNN model. When a user uploaded a document, the backend handled preprocessing (resizing, normalization, grayscale conversion) before passing it to the model for classification.

- **Model Integration**: The trained CNN model was saved in a serialized format (HDF5) and loaded within the backend. Predictions were made in real time whenever a document was submitted.

- **Frontend**: A simple, interactive interface was developed using **Streamlit**, which provided an easy-to-use web-based environment. Users could upload document images, view preprocessing outputs, and see the final prediction result (genuine or forged).

This modular architecture ensured separation of concerns, with image processing and classification handled by the backend, and interaction provided by the frontend.

System Flowchart



The process follows these steps:

1. **Upload Document** – the user uploads a scanned image or photograph of a document.

2. **Preprocessing** – the system standardizes the document (resizing, grayscale, noise reduction, thresholding).

3. **Feature Extraction & Classification** – the preprocessed image is passed to the trained CNN model, which extracts features and classifies the document.

4. **Result Display** – the system returns the result (Genuine or Forged), along with confidence scores.

This step-by-step design ensures a smooth flow of information, from input to output, while maintaining transparency for the user.

User Interface (UI)

The system interface was intentionally designed to be simple and intuitive, allowing users with little technical knowledge to interact with it.

Homepage Overview



The homepage of the system, titled **Document Forgery Detection System**, serves as the entry point for users to interact with the application. It presents a simple, intuitive, and visually appealing interface designed for ease of use. At the top of the page, a navigation bar provides access to the main sections: **Home**, **Detect Forgery**, and **About**, ensuring that users can easily move between different parts of the system.

The central banner welcomes users with the message **"Welcome to DocumentGuard"**, accompanied by a short description: *"AI-Powered System for Real-Time Document Forgery Detection"*. This immediately communicates the purpose of the application and builds user confidence in its capabilities.

The homepage layout emphasizes functionality by highlighting the core features of the system in a grid format with distinct icons. These features are:

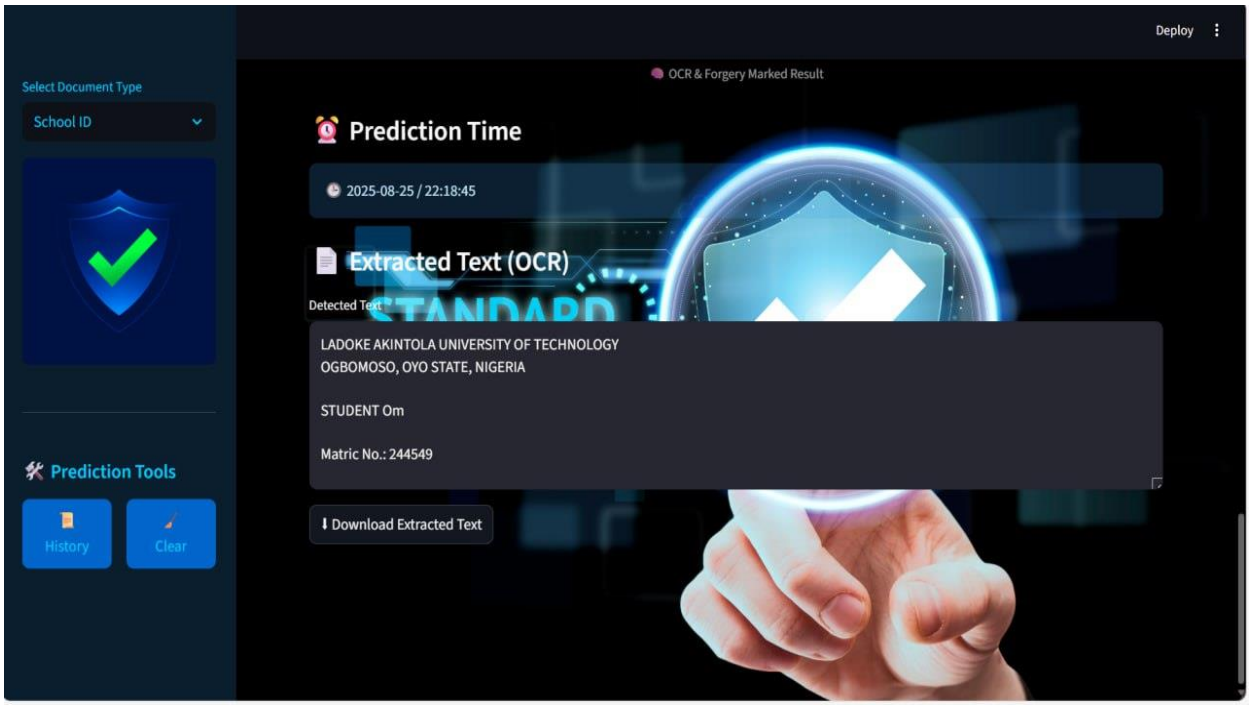
Key Functionalities Displayed on the Homepage

1. **Upload Documents**
    - Users are able to upload different types of documents for verification.

- The system supports common document formats such as **School IDs** and **National IDs**, making it adaptable for real-world institutional use.

2. **Detect Forgery**
    - This feature leverages the deep learning model integrated into the system to automatically analyze uploaded documents.
    - The AI model checks for tampering or signs of manipulation, ensuring accurate detection of forged documents.
3. **Fast Results**
    - The system provides **instant feedback** after analysis.
    - Users receive a clear verdict on whether a document is **"Original"** or **"Fake"**, minimizing delays that are common in manual verification processes.
4. **OCR Support**
    - The system is equipped with Optical Character Recognition (OCR) capabilities.
    - This allows it to extract text from scanned or uploaded documents, enabling further analysis and making the system more versatile in identifying textual inconsistencies.

- The **Detect Forgery** page is the core functional area of the system where users actively interact with the forgery detection model. The design is straightforward and user-friendly, making it accessible even for non-technical users.



On the left panel, users can **select the document type** they wish to verify, such as a *School ID* or *National ID*. This allows the system to tailor its detection process to the specific format of the document. Below this, the **Prediction Tools** section provides quick options, including a *History* button for reviewing past verification attempts and a *Clear* button to reset the interface.

At the center, the main feature is the **document upload or capture section**. Users are given two options for input:

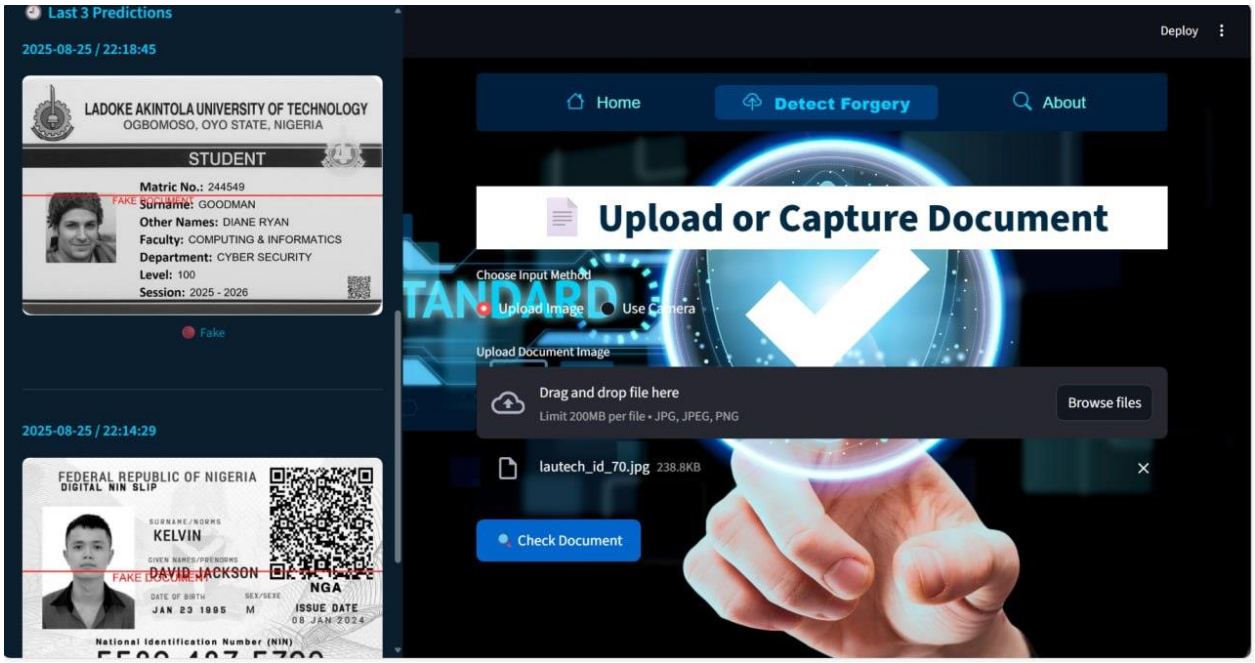
1. **Upload Image** – allows users to drag and drop files or browse their system for supported formats (JPG, JPEG, PNG) up to 200MB.

2. **Use Camera** – enables real-time document capture for instant verification.

Once a document is uploaded, the filename is displayed (e.g., *lautech_id_70.jpg*), and the user can proceed by clicking the **Check Document** button. The system then begins **image processing**, displaying a progress bar that reassures the user that analysis is ongoing.

This page represents the system's **core functionality**—linking the user's uploaded document with the deep learning model that processes and classifies it as either *original* or *forged*. By combining simplicity with functionality, the Detect Forgery page ensures smooth user experience while maintaining robust verification capability.

- **History Page**: The **History Page** provides users with a record of their recent forgery detection activities, ensuring transparency, traceability, and convenience. This feature is particularly useful for institutions and individuals who need to keep track of multiple document verification attempts over time.



On the left panel under **"Last 3 Predictions"**, the system displays thumbnails of recently verified documents along with key details such as:

- **Date and Time of Verification** – each prediction is time-stamped to help users know exactly when the analysis took place.

- **Document Preview** – a small snapshot of the uploaded document is shown, allowing users to quickly recognize and review past checks.

- **Verification Verdict** – the system clearly labels each document as *Original* or *Fake*, with a visual highlight (e.g., a red "Fake" watermark across forged IDs).

For example, in the screenshot provided, the system identified a **student ID** and a **National Identification slip** as forged documents, with both results recorded alongside their timestamps. This provides users with not only the decision but also supporting visual evidence for reference.

By maintaining a short-term history of recent checks, the page enhances user experience in several ways:

- **Quick Reference** – users can revisit recent verifications without needing to re-upload documents.
- **Verification Confidence** – timestamps and document previews help confirm that the system is functioning correctly and consistently.
- **Institutional Use** – organizations can keep a short audit trail of document verification, which can later be exported or logged for record-keeping.

Overall, the **History Page** adds value by combining usability with accountability, ensuring that the forgery detection process remains transparent and reliable for all users.

This design not only provides functionality but also builds trust in the system's decision-making by offering confidence scores.

Deployment Considerations

Although the system was tested locally, it was designed with scalability in mind. With slight modifications, it can be deployed on a **cloud environment** such as AWS or Google Cloud, allowing institutions to use it at scale. Deployment in the cloud would enable multiple users to upload documents simultaneously, with results returned in real time.

Furthermore, the modular design allows future extensions, such as integrating **OCR (Optical Character Recognition)** for text-level forgery detection or connecting with institutional databases for additional verification layers.

In summary, the system implementation demonstrates how the trained CNN model can be translated into a practical tool for forgery detection. By combining backend processing with a simple web interface, the project bridges the gap between theory and application, providing a usable solution for real-world scenarios.

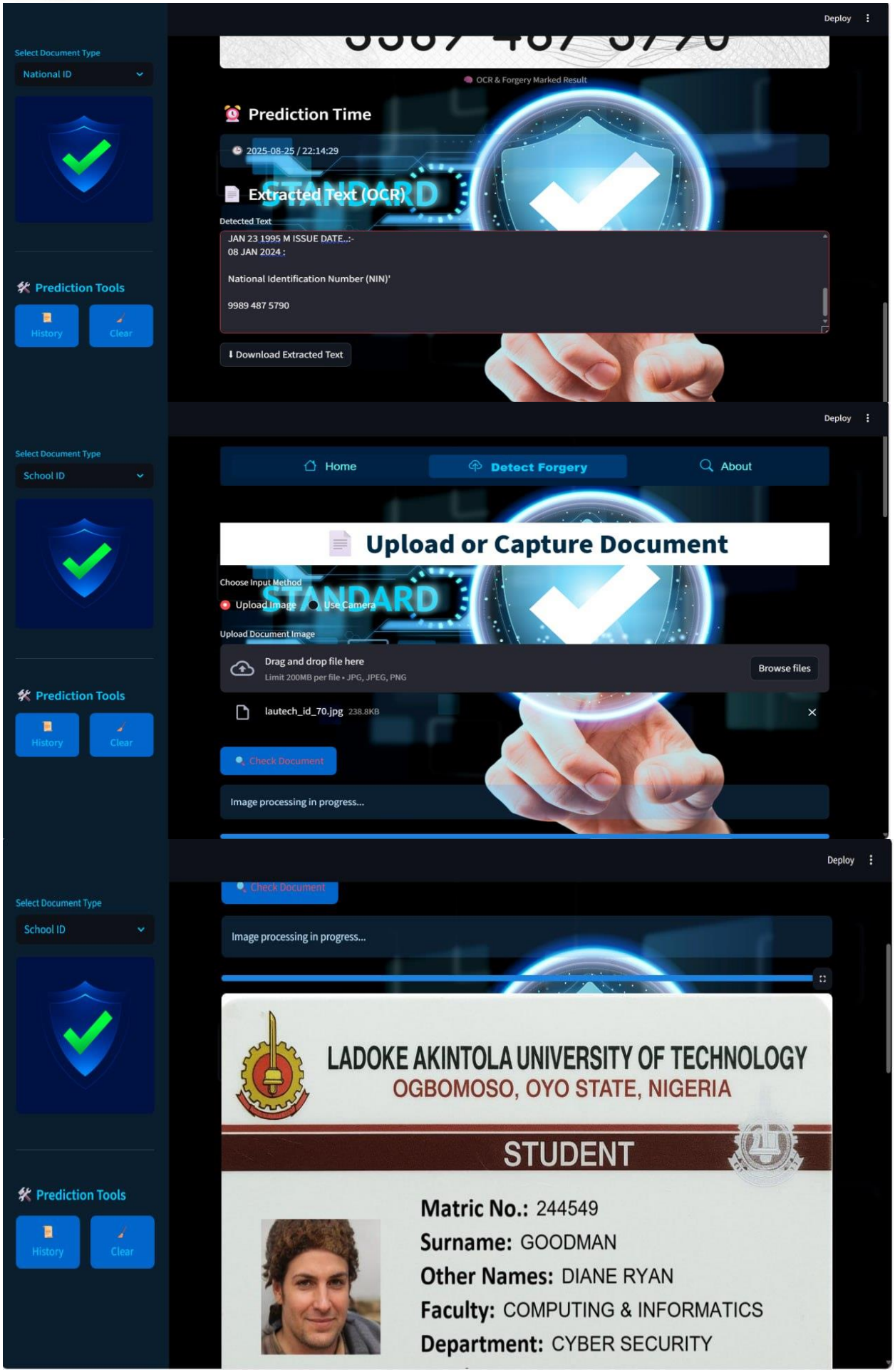
DISCUSSION OF FINDINGS

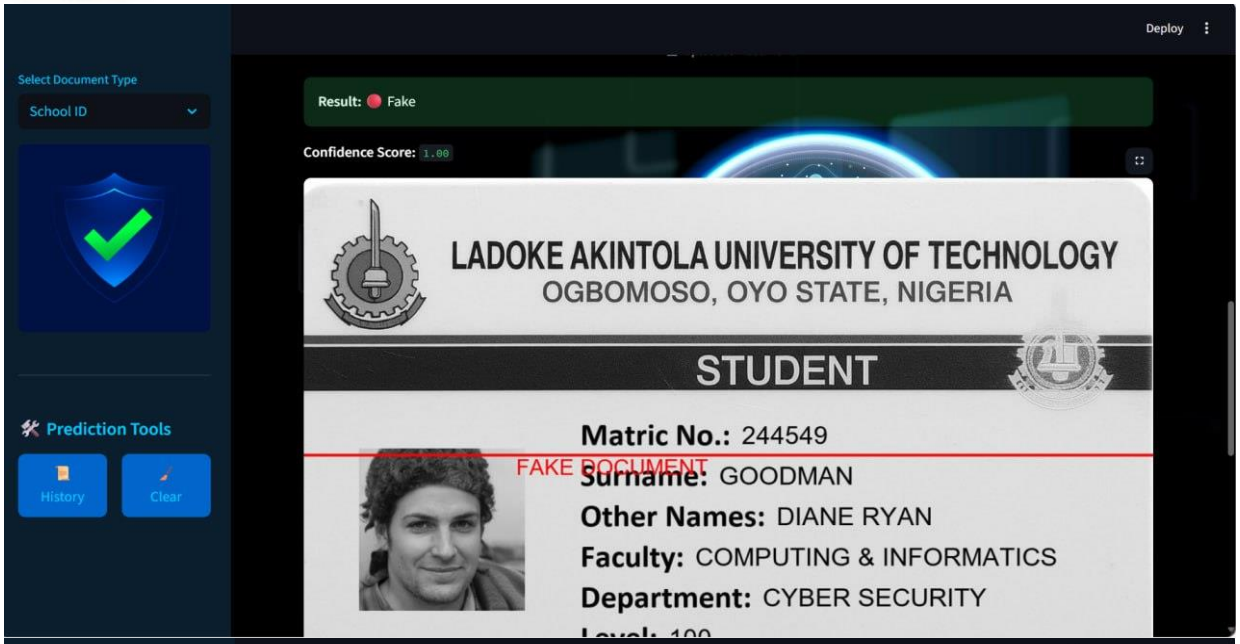
The experimental results obtained from the document forgery detection system demonstrated that the combination of image processing techniques and machine learning, particularly Convolutional Neural Networks (CNNs), is highly effective in distinguishing between genuine and forged documents. With an overall accuracy of **94%**, the system showed strong potential for real-world deployment across sectors such as education, banking, and government identity verification.

Performance Interpretation

The model's performance metrics **precision (92.5%)**, **recall (95.0%)**, and **F1-score (93.7%)** indicate a balanced and reliable classification ability. The high recall score suggests that the system excelled at identifying forged documents, which is critical in preventing fraud. In security-sensitive applications, it is preferable to flag a genuine document as suspicious (false positive) than to overlook a forgery (false negative).



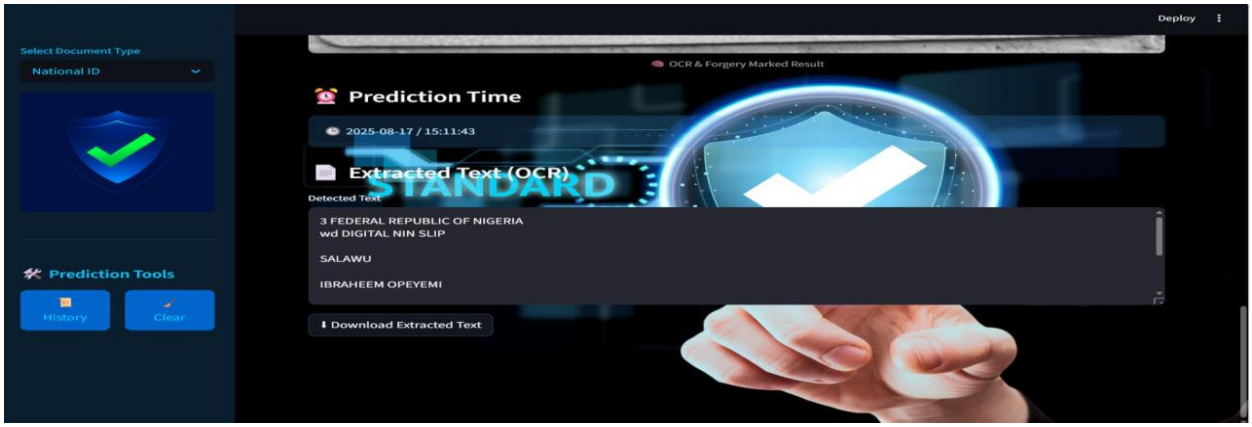

Deploy
Select Document Type
National ID
Prediction Tools
History
Clear
OCR & Forgery Marked Result
Prediction Time
2025-08-25 / 22:14:29
Extracted Text (OCR)
Detected Text
JAN 23 1995 M ISSUE DATE..:-
08 JAN 2024 :
National Identification Number (NIN)'
9989 487 5790
Download Extracted Text
Deploy
Select Document Type
School ID
Prediction Tools
History
Clear
Home
Detect Forgery
About
Upload or Capture Document
Choose Input Method
Upload Image
Use Camera
Upload Document Image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
lautech_id_70.jpg 238.8KB
Check Document
Image processing in progress...
Deploy
Select Document Type
School ID
Prediction Tools
History
Clear
Check Document
Image processing in progress...
LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY
OGBOMOSO, OYO STATE, NIGERIA
STUDENT
Matric No.: 244549
Surname: GOODMAN
Other Names: DIANE RYAN
Faculty: COMPUTING & INFORMATICS
Department: CYBER SECURITY


Deploy
Select Document Type
School ID
Result: Fake
Confidence Score: 1.00
LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY
OGBOMOSO, OYO STATE, NIGERIA
STUDENT
Matric No.: 244549
FAKE DOCUMENT
Surname: GOODMAN
Other Names: DIANE RYAN
Faculty: COMPUTING & INFORMATICS
Department: CYBER SECURITY
Prediction Tools
History
Clear


Stop
Deploy
Select Document Type
National ID
Upload or Capture Document
Choose Input Method
Upload Image
Use Camera
Upload Document Image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
9.jpg
Check Document
Image processing in progress...
FEDERAL REPUBLIC OF NIGERIA
Prediction Tools
History
Clear


Deploy
Select Document Type
National ID
FEDERAL REPUBLIC OF NIGERIA
DIGITAL NIN SLIP
SURNAME/NORMS
KELVIN
GIVEN NAMES/PRENORMS
DAVID JACKSON
DATE OF BIRTH
JAN 23 1995
SEX/SEXE
M
NGA
ISSUE DATE
08 JAN 2024
National Identification Number (NIN)
5589 487 5790
Uploaded Document
Prediction Tools
History
Clear


STANDARD
Uploaded Document
Result: Fake
Confidence Score: 1.00
FEDERAL REPUBLIC OF NIGERIA
DIGITAL NIN SLIP
SURNAME/NORMS
KELVIN
GIVEN NAMES/PRENORMS
Prediction Tools
History
Clear



However, the results also highlighted some challenges:

- **False Positives (FPs):** Genuine documents occasionally flagged as forged. These errors were often linked to low image quality, document noise, or faded printing.
- **False Negatives (FNs):** A few forgeries went undetected, particularly when modifications were subtle (e.g., minor text edits, carefully replicated signatures). These cases reveal the difficulty of detecting highly sophisticated forgeries, even with advanced machine learning models.

These findings reflect a broader challenge in forgery detection systems: the need to balance sensitivity with precision. A stricter system reduces false negatives but risks generating more false positives, while a more lenient system might allow subtle forgeries to slip through.

Comparison with Related Works

The results achieved in this project are consistent with findings from previous research. For instance, Kaur and Arora (2020) reported similar success when applying CNNs to signature forgery detection, achieving over 90% accuracy. Likewise, Sharma et al. (2021) demonstrated that combining preprocessing with deep learning significantly improves classification accuracy compared to traditional forensic analysis.

Compared to traditional methods—such as manual expert inspection or watermark verification—the proposed system demonstrated greater speed, scalability, and consistency. Manual inspection is prone to human error and bias, whereas automated systems provide objective, repeatable results. Moreover, traditional forensic approaches often require specialized expertise, while this system can be used by non-experts through its web-based interface.

Implications for Real-World Applications

The strong performance of the system suggests that it can serve as a practical tool in environments where document verification is critical:

- **Educational Institutions:** Verification of student transcripts and certificates during admissions or recruitment processes.
- **Financial Sector:** Automated verification of identity documents (e.g., ID cards, bank statements) during customer onboarding.
- **Government Agencies:** Fraud prevention in national identity schemes, passport issuance, or driver's license verification.

By reducing reliance on manual checks, the system not only improves efficiency but also enhances security against fraudulent activities.

Reflection on Model Strengths and Weaknesses

Overall, the findings demonstrate that CNN-based systems can significantly improve document forgery detection. The strengths of the model include:

- High accuracy and recall, making it dependable for detecting forgeries.
- Robustness gained through preprocessing and augmentation, enabling the system to handle varying document qualities.
- Scalability, as the model can be integrated into web-based platforms for real-time use.

Nonetheless, weaknesses were also evident:

- Sensitivity to very subtle forgeries, especially those designed to mimic genuine features closely.
- Dependence on image quality, where low-resolution or heavily compressed documents reduced accuracy.
- Limitations of dataset size, which may restrict generalization to broader document types.

In conclusion, the findings highlight the promise of deep learning–based forgery detection while also pointing to areas that require improvement. By addressing dataset diversity, incorporating more advanced architectures, and combining visual analysis with text-based verification, the system could evolve into an even more powerful and reliable tool.

LIMITATIONS OF THE SYSTEM

While the developed document forgery detection system performed strongly in terms of accuracy and reliability, it is important to acknowledge its limitations. Recognizing these constraints not only provides a balanced evaluation of the system but also highlights opportunities for future improvement.

1. Dataset Limitations

One of the most significant limitations lies in the dataset used for training and evaluation. Although balanced in terms of class distribution, the dataset was relatively small (1,000 images). A larger dataset with more varied examples would provide the model with a broader spectrum of forgery techniques and document types. Furthermore, the forged samples were synthetically created, which, while useful for training, may not fully capture the complexity and ingenuity of real-world forgeries. In practice, forgeries can involve advanced techniques such as micro-print tampering, hologram manipulation, or hybrid digital-physical edits, which were not fully represented in the dataset.

2. Model Limitations

The Convolutional Neural Network (CNN) employed was effective but not without constraints:

- **Sensitivity to subtle forgeries:** The model occasionally struggled with highly sophisticated manipulations, such as precise text edits or expertly replicated signatures, which blend seamlessly into the document.
- **Computational demand:** Training and inference required significant processing power. While feasible on the test workstation, deploying the model on lower-end devices may lead to slower processing times.
- **Risk of overfitting:** Despite augmentation techniques, the limited dataset size posed a risk of the

model memorizing patterns instead of learning generalized features. This could reduce its effectiveness when applied to unseen document types.

3. System Deployment Constraints

While the prototype system performed effectively in a controlled environment, scaling it for real-world use presents additional challenges:

- **Processing speed at scale:** Handling large volumes of documents simultaneously, such as in national ID verification systems, may strain computational resources.
- **Integration hurdles:** Deploying the system into existing institutional workflows or databases may require significant customization.
- **User trust and adoption:** For end users, transparency is essential. A system that occasionally misclassifies documents may face skepticism unless accompanied by human verification processes.

Improvements

While the developed system achieved encouraging results, there are several opportunities for enhancement to make it more robust, accurate, and applicable in real-world scenarios. These improvements focus on expanding the dataset, refining the model, and addressing practical deployment challenges.

1. Expanding and Diversifying the Dataset

A key improvement would be to assemble a larger and more diverse dataset. This includes:

- **Real-world forgeries:** Collecting actual forged documents (with appropriate ethical clearance) rather than relying solely on synthetically created samples. This would expose the model to the complexity of real forgery attempts.
- **Varied document types:** Incorporating a wider range of documents, such as passports, driver's licenses, bank statements, and land titles, would improve generalization across industries.
- **Multi-source data collection:** Gathering documents from different institutions and regions to reflect diverse formats, fonts, and security features.

A larger and more representative dataset would significantly enhance the model's ability to generalize and detect subtle or unfamiliar forgeries.

2. Adoption of Pre-Trained Deep Learning Models

While a CNN built from scratch provided strong results, leveraging **pre-trained models** (e.g., ResNet, VGGNet, EfficientNet) through transfer learning could improve accuracy further. Pre-trained networks already possess rich feature extraction capabilities learned from millions of images, allowing them to detect intricate patterns in documents with fewer training examples. Transfer learning also reduces training time and computational requirements.

3. Integration of Optical Character Recognition (OCR) and Natural Language Processing (NLP)

At present, the system focuses on visual features for forgery detection. Incorporating OCR would allow text from documents to be extracted and analyzed. This could be combined with NLP techniques to detect:

- Inconsistent text formatting or fonts.
- Suspicious changes in names, dates, or numerical values.

- Linguistic inconsistencies in certificates or transcripts.

This multimodal approach (visual + textual analysis) would greatly increase the robustness of forgery detection.

4. Improving Computational Efficiency

To reduce computational load and enable real-time deployment, improvements such as **model optimization** and **lightweight architectures** (e.g., MobileNet, TinyML) could be adopted. These models are designed for efficiency, making them suitable for deployment on low-resource environments such as mobile devices or embedded systems.

5. Cloud-Based Deployment and Scalability

For large-scale use, deploying the system in a **cloud environment** (e.g., AWS, Google Cloud, or Microsoft Azure) would allow multiple institutions to access the service simultaneously. Cloud-based deployment offers benefits such as scalability, centralized updates, and improved security. Furthermore, incorporating containerization tools (e.g., Docker, Kubernetes) would streamline integration and scaling across platforms.

6. Incorporation of Explainable AI (XAI)

One limitation of deep learning systems is their "black-box" nature. Adding explainability features would allow the system to highlight regions of the document that influenced its decision (e.g., signature areas, text regions, seals). This would build trust among users and allow human experts to verify flagged areas more easily.

7. Continuous Learning and Updating

The system should be designed to support **continuous learning**, where new document samples—particularly forged cases encountered in real-world settings—can be added to retrain and improve the model over time. This adaptive approach would ensure that the system evolves in response to emerging forgery techniques.

In conclusion, while the current system demonstrates high accuracy and strong potential, these improvements would make it more adaptable, scalable, and trustworthy. By incorporating a larger dataset, pre-trained models, OCR capabilities, and explainable AI, the system could become a comprehensive and industry-ready solution for forgery detection across multiple domains.

Comparison with Existing Methods

To better understand the strengths of the developed document forgery detection system, it is important to compare its performance and methodology with existing approaches. Over the years, several techniques have been used to detect forged documents, ranging from traditional manual inspections to rule-based digital checks and, more recently, machine learning–driven solutions.

1. Manual and Forensic Methods

Historically, document verification relied heavily on human experts such as handwriting analysts and forensic document examiners. These experts inspect physical characteristics like ink type, handwriting patterns, or embossed seals. While such methods can be highly accurate in controlled conditions, they are:

- **Time-consuming**, often requiring hours to analyze a single document.
- **Subjective**, as different experts may interpret features differently.
- **Difficult to scale**, making them unsuitable for institutions processing thousands of documents daily.

Compared to these methods, the proposed CNN-based system is **faster, more objective, and scalable**, offering automated verification in real time with consistent accuracy.

2. Rule-Based Digital Methods

Some digital systems rely on predefined rules such as watermark verification, checksum validation, or template matching. These methods are effective for detecting certain types of forgeries, such as missing watermarks or altered layouts. However, they often fail when dealing with more advanced, subtle manipulations—such as text edits or forged signatures—that do not affect the overall template.

In contrast, the developed system uses deep learning to detect **pixel-level and structural anomalies** in documents, enabling it to uncover modifications that rule-based approaches may overlook. This makes it more versatile against a wide variety of forgery techniques.

3. Traditional Image Processing Methods

Earlier computer-based approaches often relied on techniques such as edge detection, histogram analysis, or feature extraction using algorithms like Scale-Invariant Feature Transform (SIFT). While these methods provided early automation, they were limited by their reliance on handcrafted features, which do not always generalize well to complex or evolving forgery styles.

By comparison, the CNN-based model in this project automatically **learns hierarchical features** directly from the data. This eliminates the need for manual feature engineering and allows the system to adapt to different document types and forgery techniques.

4. Machine Learning and Deep Learning Approaches

Recent studies have applied machine learning models such as Support Vector Machines (SVMs), Random Forests, and CNNs for forgery detection. These approaches have demonstrated significant improvements over traditional methods. For example, Sharma et al. (2021) used CNNs for banknote authentication, achieving over 90% accuracy, while Kaur and Arora (2020) applied deep learning to signature verification with similarly high results.

The CNN developed in this project achieved **94% accuracy**, placing it on par with or above these existing efforts. Its strength lies in combining preprocessing (for noise reduction and feature clarity) with a carefully trained CNN model, which allowed it to generalize well on unseen test data.

Summary of Comparison

Methods	Strengths	Weaknesses	Proposed System Advantage
Manual/Forensic Inspection	Accurate, expert-driven	Time-consuming, subjective, not scalable	Automated, objective, scalable
Rule-Based Digital Systems	Effective for template-based verification	Easily bypassed by subtle manipulations	Detects pixel/structural anomalies
Traditional Image Processing	Simple, interpretable	Limited generalization, manual feature design	Learns features automatically
Machine Learning (ML/DL)	Higher accuracy, adaptive	Requires large datasets and computation	Achieved high accuracy (94%) with balanced dataset + augmentation

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

Introduction

This chapter brings the study to its final stage by summarizing the key aspects of the research, drawing conclusions from the findings, and presenting recommendations for practice and future work. The study set out to design and develop an image processing system for document forgery detection using Python programming language and machine learning techniques. Given the increasing threat of document forgery in critical sectors such as education, finance, and governance, the study aimed to create a system that is both reliable and applicable in real-world scenarios.

Conclusion

This study set out to address the growing problem of document forgery by developing an image processing system capable of detecting forged documents using machine learning techniques. From the onset, the primary aim was to design a solution that would not only achieve high technical accuracy but also provide practical value in real-world applications such as education, finance, and governance.

The findings of the study confirm that this aim was achieved. By combining image preprocessing techniques with a Convolutional Neural Network (CNN) model, the system successfully distinguished between genuine and forged documents with an accuracy of **94%**. The preprocessing stage proved crucial in enhancing feature clarity, while the CNN demonstrated strong generalization ability in extracting complex features for classification. The results indicated high precision and recall, ensuring both reliable detection of forgeries and minimization of false alarms.

Furthermore, the integration of the model into a user-friendly web application demonstrated the practicality of the approach. Users could upload document images and obtain real-time classification results, supported by confidence scores. This bridged the gap between research and application, showing that the system could be adopted by institutions without requiring technical expertise.

In summary, the study concluded that machine learning–based systems, when carefully designed and supported by preprocessing, offer a scalable and reliable solution to document forgery detection. While challenges remain such as dataset diversity, computational requirements, and subtle forgeries the system developed in this study represents a meaningful step toward reducing fraud and strengthening trust in official documents.

Recommendations

Based on the findings of this study, the following recommendations are proposed to guide both the practical use of the system and its future development.

1. Practical Recommendations

- **Adoption by Institutions:** Educational institutions, banks, and government agencies should consider adopting forgery detection systems to strengthen verification processes. Doing so will reduce dependence on manual inspections, minimize human errors, and speed up decision-making.

- **Integration into Existing Workflows:** The system should be integrated into existing databases and verification processes rather than functioning as a standalone tool. For instance, linking with admission portals, banking KYC systems, or government ID registries would enhance its usefulness.

- **Complementary Use with Human Oversight:** While the system is highly accurate, it should be used

as a supportive tool alongside human verification experts. Cases flagged as suspicious could be reviewed by experts for final confirmation, ensuring both efficiency and accuracy.

- **Awareness and Training:** Institutions implementing the system should provide basic training for staff to understand its use and limitations, as well as to interpret confidence scores and flagged results correctly.

2. Technical Recommendations

- **Dataset Expansion:** Future versions of the system should be trained on larger, more diverse datasets that include a wider variety of document types and real-world forgeries. This would improve the model's ability to generalize across different use cases.
- **Use of Pre-Trained Models:** Developers should explore transfer learning with advanced pre-trained architectures (e.g., ResNet, EfficientNet, Transformer-based models) to improve accuracy and reduce training time.
- **Incorporation of OCR and NLP:** To enhance detection of text-level forgeries, Optical Character Recognition (OCR) combined with Natural Language Processing (NLP) should be integrated, enabling the system to analyze both visual and textual inconsistencies.
- **Deployment on Cloud Platforms:** To ensure scalability and accessibility, future systems should be deployed on cloud infrastructures, enabling institutions to process large volumes of documents simultaneously without hardware limitations.
- **Explainable AI Integration:** Adding interpretability features would allow the system to highlight specific areas of a document (e.g., signature, seal, or altered text) that influenced its decision. This would increase transparency and build user trust.
- **Continuous Learning:** The system should be designed to update itself with new forgery patterns through continuous retraining. This adaptive feature will ensure relevance as forgery techniques evolve.

In essence, the study recommends that while the developed system already shows strong performance, its adoption should be coupled with institutional strategies for fraud prevention, and its future development should focus on enhancing scalability, robustness, and interpretability.

SUGGESTIONS FOR FUTURE WORK

Although the system developed in this study achieved promising results, there is room for further improvement and exploration. Future research and development efforts could focus on the following areas:

1. Expansion of Dataset Sources

Future work should focus on collecting larger and more representative datasets. This should include real-world forged documents rather than relying heavily on synthetically generated samples. Expanding the dataset to cover multiple document categories such as passports, driver's licenses, land titles, and academic certificates would help improve the system's adaptability across industries.

2. Hybrid Detection Approaches

Combining image-based analysis with other techniques could strengthen detection. For example, integrating **cryptographic verification (e.g., digital signatures, watermarks)** with machine learning could provide multi-layered protection against forgery. Such hybrid systems would be harder for forgers to bypass.

3. Advanced Deep Learning Architectures

While the CNN model was effective, future studies could explore more advanced models such as **Vision Transformers (ViT)** or **Generative Adversarial Networks (GANs)** for enhanced forgery detection. GANs, for instance, could be used both to simulate more realistic forgeries for training and to detect sophisticated manipulations.

4. Integration with Text-Based Analysis

Future systems should extend beyond visual features by incorporating **Optical Character Recognition (OCR)** and **Natural Language Processing (NLP)**. This would enable the system to analyze textual content for inconsistencies, such as mismatched dates, altered names, or linguistic irregularities, complementing the visual detection.

5. Real-Time Mobile and Cloud Deployment

Further research should explore deploying the system in real-time mobile applications. Lightweight models such as MobileNet could allow users to verify documents instantly using smartphones. Cloud-based versions could also provide scalable, on-demand services for large institutions.

6. Explainability and User Trust

Explainable AI (XAI) should be incorporated into future systems to make model decisions transparent. For example, highlighting regions of a document where tampering is suspected would increase user trust and help forensic experts validate results more easily.

7. Continuous Learning Frameworks

Forgers constantly evolve their methods, making static detection systems less effective over time. Future work could focus on **incremental or online learning approaches**, allowing the system to update its knowledge base as new forgery techniques are identified.

8. Cross-Language and Cross-Cultural Adaptability

Since documents vary across regions and languages, future research should also consider building systems that are adaptable across multiple formats, languages, and security features. This would make forgery detection systems globally applicable.

Conclusion

This study set out to develop an image processing system for document forgery detection using Python and machine learning techniques. Through the design of a CNN-based model, supported by careful preprocessing and integration into a user-friendly web application, the system demonstrated strong potential in tackling the persistent issue of document forgery. The results revealed high accuracy and reliability, proving that deep learning methods can offer scalable and practical solutions where traditional approaches fall short.

The research also emphasized that while the system is effective, challenges such as dataset limitations, computational demands, and evolving forgery methods must be addressed to ensure its long-term relevance. To this end, practical recommendations and suggestions for future work were proposed, including dataset expansion, the integration of OCR and NLP, the use of pre-trained models, cloud deployment, and the incorporation of explainable AI.

In conclusion, this project not only contributes to academic research in the field of document forensics but also provides a practical framework that can be adapted for real-world applications in education, finance, government, and beyond. By continuing to build on the foundation laid in this study, the system has the potential to evolve into a robust and widely adopted tool in the global fight against document forgery.

FRONTEND CODE

```
import streamlit as st
import cv2
import numpy as np
import time
import os
import io
import base64
from PIL import Image
from datetime import datetime
from streamlit_option_menu import option_menu
# Import helpers
from generator import (
    preprocess_image,
    extract_text,
    model,
    model2,
    mark_fake_document,
)
# ---------------- Streamlit Config ----------------
st.set_page_config(
    page_title="Document Forgery Detection",
    page_icon="📎",
    layout="wide",
    initial_sidebar_state="expanded"
)
st.markdown("""
<style>
.stApp {
```

```
    background-color: #010a14;
}
[data-testid="stSidebar"] {
    background-color: #0b1e2d;
}
[data-testid="stSidebar"] * {
    color: #00ccff;
}
@media only screen and (max-width: 768px) {
    .custom-header {
        font-size: 26px !important;
        padding: 6px !important;
        margin-bottom: 15px;
        word-wrap: break-word !important;
        overflow-wrap: break-word !important;
        white-space: normal !important;
        display: block !important;
    }
    .welcome-box, .card {
        padding: 15px !important;
        font-size: 16px !important;
        word-wrap: break-word !important;
        overflow-wrap: break-word !important;
        white-space: normal !important;
        display: block !important;
    }
    .stButton button, .stRadio > div {
        font-size: 15px !important;
    }
```

```
.element-container {
    padding-left: 10px !important;
    padding-right: 10px !important;
}
.block-container {
    padding: 1rem !important;
}
.stImage img {
    width: 100% !important;
    height: auto !important;
}
.nav-link {
    font-size: 14px !important;
    padding: 5px 8px !important;
}
.stTextArea textarea {
    font-size: 14px !important;
}
.css-1d391kg {
    position: fixed !important;
    top: 0;
    left: 0;
    right: 0;
    z-index: 9999;
}
.block-container {
    padding-top: 100px !important;
}
.welcome-box h1, .welcome-box h2, .welcome-box p {
```

```
        word-break: break-word !important;
        overflow-wrap: break-word !important;
        white-space: normal !important;
        display: block !important;
    }
}
@media only screen and (min-width: 769px) {
    .css-1d391kg {
        position: sticky !important;
        top: 0;
        z-index: 9999;
    }
    .block-container {
        padding-top: 60px !important;
    }
    .welcome-box h1, .welcome-box h2, .welcome-box p {
        word-break: break-word !important;
        overflow-wrap: break-word !important;
        white-space: normal !important;
        display: block !important;
    }
}
</style>
""", unsafe_allow_html=True)
# ---------- Background / Styling ----------
def set_background(image_path: str):
    with open(image_path, "rb") as image_file:
        encoded = base64.b64encode(image_file.read()).decode()
    st.markdown(
```

```
        f"""
        <style>
        .stApp {{
            background-image: url("{encoded}");
            background-size: cover;
            background-attachment: fixed;
        }}
        </style>
        """,
        unsafe_allow_html=True
    )
set_background("image/new.jpg")
# ---------------- Navigation ----------------
selected = option_menu(
    menu_title=None,
    options=["Home", "Detect Forgery", "About"],
    icons=["house", "search", "info-circle"],
    default_index=0,
    orientation="horizontal",
    styles={
        "container": {"background-color": "#001F3F"},
        "icon": {"color": "#00CFFF", "font-size": "20px"},
        "nav-link": {
            "font-size": "18px",
            "color": "#66FCF1",
            "margin": "5px",
            "--hover-color": "#003B73",
        },
        "nav-link-selected": {
```

```
            "background-color": "#003B73",
            "color": "#00CFFF",
        },
    },
)
# ---------- Header ----------
def header():
    st.markdown(
        """
        <style>
        .custom-header {
            font-size: 40px;
            font-weight: bold;
            color: #1f4e79;
            text-align: center;
            padding: 10px;
            border-bottom: 3px solid #1f4e79;
            margin-bottom: 25px;
            background-color: rgba(255,255,255,0.85);
        }
        </style>
        <div class="custom-header">🛡️ Document Forgery Detection System</div>
        """,
        unsafe_allow_html=True,
    )
# ---------- Home ----------
if selected == 'Home':
    header()
    st.markdown("""
```

```
<style>
.home-container {
    background-color: rgba(0, 31, 63, 0.9);
    border-radius: 20px;
    padding: 30px;
    color: #E0F7FA;
    box-shadow: 0 0 15px rgba(0, 255, 255, 0.2);
    margin-top: 20px;
    font-family: 'Segoe UI', sans-serif;
    max-width: 100%;
    box-sizing: border-box;
}
.home-title {
    text-align: center;
    font-size: 2.5em;
    font-weight: bold;
    color: #00CFFF;
    margin-bottom: 20px;
}
.home-subtitle {
    text-align: center;
    font-size: 1.2em;
    color: #B2EBF2;
    margin-bottom: 30px;
}
.feature-grid {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
```

```
    gap: 20px;
}
.feature-card {
    flex: 1 1 200px;
    max-width: 250px;
    background-color: #012742;
    border-radius: 12px;
    padding: 20px;
    text-align: center;
    transition: transform 0.2s ease;
    box-sizing: border-box;
}
.feature-card:hover {
    transform: scale(1.05);
}
.feature-icon {
    font-size: 2em;
    color: #66FCF1;
    margin-bottom: 10px;
}
.feature-text {
    font-size: 1.05em;
    color: #ffffff;
}
/* 🔧 Responsive Fixes */
@media (max-width: 768px) {
    .home-title {
        font-size: 1.8em;
    }
```

```
        .home-subtitle {
            font-size: 1em;
        }
        .feature-grid {
            flex-direction: column;
            align-items: center;
        }
        .feature-card {
            width: 90%;
            max-width: none;
        }
    }
    </style>
    <div class="home-container">
        <div class="home-title">👋 Welcome to DocumentGuard</div>
        <div class="home-subtitle">
            AI-Powered System for Real-Time Document Forgery Detection
        </div>
        <div class="feature-grid">
            <div class="feature-card">
                <div class="feature-icon">📤</div>
                <div class="feature-text"><b>Upload Documents</b><br>Supported formats like School ID & National ID</div>
            </div>
            <div class="feature-card">
                <div class="feature-icon">🔍</div>
                <div class="feature-text"><b>Detect Forgery</b><br>Deep Learning model checks for tampering</div>
            </div>
            <div class="feature-card">
```

```
            <div class="feature-icon">⚡</div>
            <div class="feature-text"><b>Fast Results</b><br>Instant verdict: Original ✅ or Fake ❌</div>
        </div>
        <div class="feature-card">
            <div class="feature-icon">📝</div>
            <div class="feature-text"><b>OCR Support</b><br>Extract text from documents using AI</div>
        </div>
      </div>
    </div>
    """, unsafe_allow_html=True)
# Detect Forgery
elif selected == 'Detect Forgery':
    if 'history' not in st.session_state:
        st.session_state.history = []
    if 'show_history' not in st.session_state:
        st.session_state.show_history = False
    # Sidebar: Document type, branding image, and tools
    with st.sidebar:
        doc_type = st.selectbox("Select Document Type", options=['National ID', 'School ID'], key='doc_type')
        st.image('image/new2.jpg', use_container_width=True)
        st.divider()
        st.markdown("## 🛠️ Prediction Tools")
        col1, col2 = st.columns(2)
        with col1:
            if st.button("📜 History"):
                st.session_state.show_history = not st.session_state.show_history
        with col2:
```

```
        if st.button("□ Clear"):
            st.session_state.history = []
            st.session_state.show_history = False
            st.rerun()
    if st.session_state.show_history:
        st.markdown("---")
        st.markdown("### 🕘 Last 3 Predictions")
        if st.session_state.history:
            for record in reversed(st.session_state.history[-3:]):
                st.markdown(f"**{record['timestamp']}**")
                img_bytes = base64.b64decode(record["image"])
                image = Image.open(io.BytesIO(img_bytes))
                st.image(image, caption=record["result"], use_container_width=True)
                st.markdown("---")
        else:
            st.info("No predictions yet.")
# Style block
st.markdown("""
    <style>
        .header-title {
            text-align: center;
            font-size: 2.7em;
            font-weight: 800;
            margin-top: 20px;
            color: #002b45;
            background-color: white;
        }
        # .upload-box {
        #     background-color: #f0f4f8;
```

```
        #    padding: 30px;
        #    border-radius: 14px;
        #    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.06);
        #    margin-top: 20px;
        }
        .stButton>button {
           background-color: #0066cc;
           color: white;
           border: none;
           padding: 10px 20px;
           border-radius: 8px;
           font-weight: bold;
           transition: background-color 0.3s;
        }
        .stButton>button:hover {
           background-color: #004d99;
        }
      </style>
      <div class='header-title'>📄 Upload or Capture Document</div>
   """, unsafe_allow_html=True)
   # Input method
   with st.container():
      st.markdown("<div class='upload-box'>", unsafe_allow_html=True)
      input_method = st.radio("Choose Input Method", ["Upload Image", "Use Camera"], horizontal=True)
      uploaded_file = st.file_uploader("Upload Document Image", type=["jpg", "jpeg", "png"]) if input_method == "Upload Image" else None
      camera_image = st.camera_input("Take a picture") if input_method == "Use Camera" else None
      st.markdown("</div>", unsafe_allow_html=True)
   image_input = uploaded_file or camera_image
```

```
if image_input is not None and st.button('🔍 Check Document'):
    st.info('Image processing in progress...')
    progress_bar = st.progress(0)
    for percent in range(100):
        time.sleep(0.01)
        progress_bar.progress(percent + 1)
    # Convert and process image
    file_bytes = np.asarray(bytearray(image_input.read()), dtype=np.uint8)
    original_cv2 = cv2.imdecode(file_bytes, 1)
    gray_cv2 = cv2.cvtColor(original_cv2, cv2.COLOR_BGR2GRAY)
    gray_display = cv2.cvtColor(gray_cv2, cv2.COLOR_GRAY2RGB)
    img = Image.fromarray(gray_display)
    img.save('output.jpg')
    input_img, original_cv2 = preprocess_image(img)
    # Use different model based on document type
    if doc_type == 'National ID':
        prediction = model.predict(input_img)[0][0]
    else:
        prediction = model2.predict(input_img)[0][0]
    is_fake = prediction < 0.5
    label = "⚫ Fake" if is_fake else "□ Original"
    confidence = (1 - prediction) if is_fake else prediction
    # Display results
    st.image(image_input, caption="📤 Uploaded Document", use_container_width=True)
    st.success(f"**Result:** {label}")
    st.markdown(f"**Confidence Score:** `{confidence:.2f}`")
    prediction_time = datetime.now().strftime("%Y-%m-%d / %H:%M:%S")
    ocr_img = mark_fake_document("output.jpg", is_fake)
    st.image(ocr_img, caption="□ OCR & Forgery Marked Result", use_container_width=True)
```

```
        st.subheader("⏰ Prediction Time")
        st.info(f"🕒 {prediction_time}")
        st.subheader("📄 Extracted Text (OCR)")
        text = extract_text(original_cv2)
        st.text_area("Detected Text", text, height=200)
        st.download_button("⬇ Download Extracted Text", text, file_name="ocr_output.txt")
        img_buffer = io.BytesIO()
        ocr_img.save(img_buffer, format='JPEG')
        img_base64 = base64.b64encode(img_buffer.getvalue()).decode('utf-8')
        st.session_state.history.append({"timestamp": prediction_time, "image": img_base64, "result": label})
        os.remove("output.jpg")
elif selected == 'About':
    st.markdown("""
    <style>
        .about-container {
            background-color: #f7f9fa;
            padding: 40px;
            border-radius: 16px;
            font-family: 'Segoe UI', sans-serif;
            color: #333333;
            margin-top: 30px;
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
        }
        .about-header {
            text-align: center;
            font-size: 2.5em;
            font-weight: 700;
            color: #003366;
```

```
    margin-bottom: 10px;
}
.about-subtitle {
    text-align: center;
    font-size: 1.2em;
    color: #666;
    margin-bottom: 40px;
}
.section {
    margin-bottom: 40px;
}
.section h3 {
    color: #005580;
    margin-bottom: 15px;
    border-bottom: 2px solid #e0e0e0;
    padding-bottom: 8px;
}
.section p, .section ul {
    font-size: 1.05em;
    line-height: 1.6;
}
.section ul {
    padding-left: 20px;
}
.section ul li {
    margin-bottom: 10px;
}
.developer-card {
    background-color: #e6f2ff;
```

```
        padding: 20px;
        border-radius: 12px;
        margin-top: 30px;
    }
    .developer-card h4 {
        margin-bottom: 5px;
        color: #004d80;
    }
    .contact-info a {
        color: #0066cc;
        text-decoration: none;
    }
    .footer-note {
        text-align: center;
        font-size: 0.95em;
        color: #999999;
        margin-top: 30px;
    }
    @media (max-width: 768px) {
        .about-container {
            padding: 20px;
        }
        .about-header {
            font-size: 2em;
        }
    }
</style>
<div class="about-container">
    <div class="about-header">DocumentGuard: Forgery Detection System</div>
```

AI-Powered Document Validation Tool

<h3>Overview</h3>

DocumentGuard helps individuals and institutions verify the authenticity of identity documents like School IDs and National IDs using AI-based image processing and computer vision models.

<h3>Core Features</h3>

📷 Upload or use camera to submit document image

🤖 Detect forged or altered regions using CNN models

🔤 Use OCR to extract embedded text

📊 View prediction scores and confidence levels

🗂️ Review previous scans in the history panel

🖼️ Visual result overlays for transparency

<h3>How It Works</h3>

Image is resized and preprocessed (grayscale, normalization)

Convolutional Neural Network (CNN) checks for tampering or forgery

Results are displayed with textual extraction and image annotation

History is logged for traceability and easy reference

<h3>Developer</h3>

```
        <h4>Olarinde Olateju Rachael</h4>
        <p>Data Scientist | Python Programmer | Streamlit Developer</p>
        <p>📍 Ogbomoso, Oyo State, Nigeria</p>
      </div>
      <div class="section contact-info">
        <h3>Contact</h3>
        <ul>
          <li>📧 Email: <a href="mailto:olarindeolatejur@gmail.com">olarindeolatejur@gmail.com</a></li>
          <li>💻 GitHub: <a href="https://github.com/Olateju" target="_blank">github.com/Olateju</a></li>
          <li>🔗 LinkedIn: <a href="https://linkedin.com/in/Olateju" target="_blank">linkedin.com/in/Olateju</a></li>
        </ul>
      </div>
      <div class="footer-note">
        © 2025 DocumentGuard | Built using Streamlit and Python
      </div>
    </div>
    """, unsafe_allow_html=True)
        <p>📍 Ogbomoso, Oyo State, Nigeria</p>
      </div>
      """,
      unsafe_allow_html=True,
    )
```

GENERATOR

```
import os
import cv2
import gdown
import numpy as np
```

```
import easyocr
from PIL import Image, ImageDraw, ImageFont
from tensorflow.keras.models import load_model
import streamlit as st
from transformers import pipeline
# ------------------------------
# Google Drive File IDs
# ------------------------------
DOCUMENT_MODEL_ID = "1VTzUIddlLttbLAZhldzI8GMQsrvrhG7-"
SCHOOL_MODEL_ID = "12UHOLzK2tb8BaMXoTVzV00kOpixjOz5C"
# ------------------------------
# Helper: download from Drive
# ------------------------------
def download_from_drive(file_id, output):
    if not os.path.exists(output):
        url = f"https://drive.google.com/uc?id={file_id}"
        gdown.download(url, output, quiet=False)
# ------------------------------
# Load models (cached)
# ------------------------------
@st.cache_resource
def load_forgery_model():
    model_path = "document_forgery_model.h5"
    download_from_drive(DOCUMENT_MODEL_ID, model_path)
    return load_model(model_path)
@st.cache_resource
def load_school_model():
    model_path = "school_forgery_model.h5"
    download_from_drive(SCHOOL_MODEL_ID, model_path)
```

```
    return load_model(model_path)
model = load_forgery_model()
model2 = load_school_model()
# Hugging Face document classifier (lightweight text/image model)
@st.cache_resource
def load_document_classifier():
    return pipeline("image-classification", model="microsoft/dit-base")
doc_classifier = load_document_classifier()
# ------------------------------
# Preprocessing
# ------------------------------
def preprocess_image(image):
    """Prepare image for forgery models."""
    img = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
    resized = cv2.resize(img, (224, 224))
    normalized = resized / 255.0
    reshaped = np.reshape(normalized, (1, 224, 224, 3))
    return reshaped, img
@st.cache_resource
def preprocess_for_ocr(image_path):
    """Preprocess image for OCR (resize + grayscale)."""
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, (800, 800))
    return resized
# ------------------------------
# OCR + Text extraction
# ------------------------------
def extract_text(image_input):
```

```
    """Extract text using EasyOCR."""
    ocr = easyocr.Reader(['en'])
    result = ocr.readtext(image_input)
    text = " ".join([res[1] for res in result])
    return text.strip()
# ------------------------------
# Fake document marking
# ------------------------------
def mark_fake_document(image_path, is_fake):
    img = Image.open(image_path).convert("RGB")
    draw = ImageDraw.Draw(img)
    width, height = img.size
    if is_fake:
        try:
            font = ImageFont.truetype("DejaVuSans.ttf", size=40)
        except:
            font = ImageFont.load_default()
        draw.text(
            (width // 4, height // 2),
            "FAKE DOCUMENT",
            fill="red",
            font=font
        )
        draw.line((0, height // 2, width, height // 2), fill="red", width=5)
    return img
# ------------------------------
# Document validation
# ------------------------------
# def is_document(image):
```

```
#     """
#     Validate if uploaded image is a document.
#     Uses contour geometry + OCR + Hugging Face classifier.
#     """
#     # Convert to grayscale
#     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#     # Step 1: Contour check (documents are rectangles)
#     edges = cv2.Canny(gray, 50, 150)
#     contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#     for cnt in contours:
#         approx = cv2.approxPolyDP(cnt, 0.02 * cv2.arcLength(cnt, True), True)
#         if len(approx) == 4:  # rectangle
#             x, y, w, h = cv2.boundingRect(approx)
#             aspect_ratio = w / float(h)
#             if 0.5 < aspect_ratio < 2.0 and w > 200 and h > 200:
#                 break
#     else:
#         return False
#     # Step 2: OCR density check
#     ocr = easyocr.Reader(['en'])
#     result = ocr.readtext(gray)
#     if len(result) < 3:  # must have some text
#         return False
#     # Step 3: Hugging Face check (classifier confidence)
#     pil_img = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
#     predictions = doc_classifier(pil_img)
#     top_label = predictions[0]["label"].lower()
#     confidence = predictions[0]["score"]
#     if "document" in top_label and confidence > 0.7:
```

```
#         return True
#     return False
```

NATIONAL IDENTIFICATION CARD MODEL

```
import os
import cv2
import gdown
import numpy as np
import easyocr
from PIL import Image, ImageDraw, ImageFont
from tensorflow.keras.models import load_model
import streamlit as st
from transformers import pipeline
# ------------------------------
# Google Drive File IDs
# ------------------------------
DOCUMENT_MODEL_ID = "1VTzUIddlLttbLAZhldzI8GMQsrvrhG7-"
SCHOOL_MODEL_ID = "12UHOLzK2tb8BaMXoTVzV00kOpixjOz5C"
# ------------------------------
# Helper: download from Drive
# ------------------------------
def download_from_drive(file_id, output):
    if not os.path.exists(output):
        url = f"https://drive.google.com/uc?id={file_id}"
        gdown.download(url, output, quiet=False)
# ------------------------------
# Load models (cached)
# ------------------------------
@st.cache_resource
def load_forgery_model():
```

```
    model_path = "document_forgery_model.h5"
    download_from_drive(DOCUMENT_MODEL_ID, model_path)
    return load_model(model_path)
@st.cache_resource
def load_school_model():
    model_path = "school_forgery_model.h5"
    download_from_drive(SCHOOL_MODEL_ID, model_path)
    return load_model(model_path)
model = load_forgery_model()
model2 = load_school_model()
# Hugging Face document classifier (lightweight text/image model)
@st.cache_resource
def load_document_classifier():
    return pipeline("image-classification", model="microsoft/dit-base")
doc_classifier = load_document_classifier()
# ------------------------------
# Preprocessing
# ------------------------------
def preprocess_image(image):
    """Prepare image for forgery models."""
    img = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
    resized = cv2.resize(img, (224, 224))
    normalized = resized / 255.0
    reshaped = np.reshape(normalized, (1, 224, 224, 3))
    return reshaped, img
@st.cache_resource
def preprocess_for_ocr(image_path):
    """Preprocess image for OCR (resize + grayscale)."""
    image = cv2.imread(image_path)
```

```
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, (800, 800))
    return resized
# ------------------------------
# OCR + Text extraction
# ------------------------------
def extract_text(image_input):
    """Extract text using EasyOCR."""
    ocr = easyocr.Reader(['en'])
    result = ocr.readtext(image_input)
    text = " ".join([res[1] for res in result])
    return text.strip()
# ------------------------------
# Fake document marking
# ------------------------------
def mark_fake_document(image_path, is_fake):
    img = Image.open(image_path).convert("RGB")
    draw = ImageDraw.Draw(img)
    width, height = img.size
    if is_fake:
        try:
            font = ImageFont.truetype("DejaVuSans.ttf", size=40)
        except:
            font = ImageFont.load_default()
        draw.text(
            (width // 4, height // 2),
            "FAKE DOCUMENT",
            fill="red",
            font=font
```

```
        )
        draw.line((0, height // 2, width, height // 2), fill="red", width=5)
    return img
# ------------------------------
# Document validation
# ------------------------------
# def is_document(image):
#     """
#     Validate if uploaded image is a document.
#     Uses contour geometry + OCR + Hugging Face classifier.
#     """
#     # Convert to grayscale
#     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#     # Step 1: Contour check (documents are rectangles)
#     edges = cv2.Canny(gray, 50, 150)
#     contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#     for cnt in contours:
#         approx = cv2.approxPolyDP(cnt, 0.02 * cv2.arcLength(cnt, True), True)
#         if len(approx) == 4:  # rectangle
#             x, y, w, h = cv2.boundingRect(approx)
#             aspect_ratio = w / float(h)
#             if 0.5 < aspect_ratio < 2.0 and w > 200 and h > 200:
#                 break
#     else:
#         return False
#     # Step 2: OCR density check
#     ocr = easyocr.Reader(['en'])
#     result = ocr.readtext(gray)
#     if len(result) < 3:  # must have some text
```

```
#         return False
#     # Step 3: Hugging Face check (classifier confidence)
#     pil_img = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
#     predictions = doc_classifier(pil_img)
#     top_label = predictions[0]["label"].lower()
#     confidence = predictions[0]["score"]
#     if "document" in top_label and confidence > 0.7:
#         return True
#     return False
```

SCHOOL IDENTIFICATION CARD MODEL

```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import class_weight
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.applications import VGG16
from tensorflow.keras.metrics import Precision, Recall
# === Configuration ===
base_dir = "data"
train_dir = os.path.join(base_dir, "school_id")
img_height, img_width = 224, 224
batch_size = 32
num_epochs = 50
fine_tune_epochs = 20
val_split = 0.2
class_names = ['forged', 'original']
# === Data Augmentation and Preparation ===
```

```
def prepare_data(train_dir, img_height, img_width, batch_size, val_split):
    datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        zoom_range=0.2,
        width_shift_range=0.1,
        height_shift_range=0.1,
        brightness_range=[0.8, 1.2],
        shear_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest',
        validation_split=val_split
    )
    train_data = datagen.flow_from_directory(
        train_dir,
        target_size=(img_height, img_width),
        classes=class_names,
        batch_size=batch_size,
        class_mode='binary',
        subset='training',
        shuffle=True
    )
    val_data = datagen.flow_from_directory(
        train_dir,
        target_size=(img_height, img_width),
        classes=class_names,
        batch_size=batch_size,
        class_mode='binary',
        subset='validation',
```

```
        shuffle=False
    )
    return train_data, val_data
# === Compute Class Weights ===
def compute_class_weights(train_data):
    labels = train_data.classes
    return dict(enumerate(class_weight.compute_class_weight(
        class_weight='balanced',
        classes=np.unique(labels),
        y=labels
    )))
# === Model Definition ===
def create_model(input_shape):
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)
    base_model.trainable = False  # Freeze the convolutional base
    model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
        layers.Dropout(0.5),  # Increased dropout rate
        layers.Dense(1, activation='sigmoid')  # Binary classification
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy',
                  metrics=['accuracy', Precision(), Recall()])
    return model
# === Callbacks ===
def get_callbacks():
    checkpoint_cb = ModelCheckpoint(
        "document1_forgery_model.keras",
```

```
        monitor="val_accuracy",
        save_best_only=True,
        verbose=1
    )
    earlystop_cb = EarlyStopping(
        monitor="val_accuracy",
        patience=5,
        restore_best_weights=True,
        verbose=1
    )
    lr_scheduler = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=2,
        min_lr=1e-6
    )
    return [checkpoint_cb, earlystop_cb, lr_scheduler]
# === Visualization Function ===
def plot_history(history, fine_tune=False):
    plt.figure(figsize=(10, 5))
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Model Accuracy over Epochs' + (' (Fine-Tuning)' if fine_tune else ''))
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
# === Main Execution ===
train_data, val_data = prepare_data(train_dir, img_height, img_width, batch_size, val_split)
class_weights = compute_class_weights(train_data)
# Create and train the model
model = create_model((img_height, img_width, 3))
callbacks = get_callbacks()
# Train the model
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=num_epochs,
    class_weight=class_weights,
    callbacks=callbacks
)
# === Fine-tuning the Model ===
# Unfreeze some layers for fine-tuning
for layer in model.layers[-4:]:
    layer.trainable = True
# Recompile the model with a lower learning rate
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', Precision(), Recall()])
# Continue training with fine-tuning
history_fine_tune = model.fit(
    train_data,
    validation_data=val_data,
    epochs=fine_tune_epochs,
    class_weight=class_weights,
    callbacks=callbacks
)
# Plot training history
```

plot_history(history)

plot_history(history_fine_tune, fine_tune=True)

REFERENCES

1. **Jain, A. K., Griess, F. D., & Connell, S. D. (2002).** "On-line signature verification." *Pattern Recognition*, 35(12), 2963–2972.
   (For discussions on signature and handwritten forgery detection.)
2. **Nikam, S. B., & Agarwal, S. (2015).** "Document Forgery Detection Using Image Processing Techniques." *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(4), 1–5.
   (For general document forgery detection using image processing.)
3. **Gonzalez, R. C., & Woods, R. E. (2018).** *Digital Image Processing* (4th ed.). Pearson.
   (A foundational book on image processing techniques like thresholding, edge detection, etc.)
4. **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning*. MIT Press.
   (For concepts related to Convolutional Neural Networks (CNNs) and machine learning.)
5. **Kaur, L., & Arora, A. (2020).** "A Reviewof Document Forgery Detection Techniques." *International Journal of Computer Applications*, 176(30), 1–6.
   (Covers an overview of existing forgery detection methods.)
6. **OpenCV Documentation.** (n.d.). https://docs.opencv.org/
   (For practical implementations of image processing methods.)
7. **TensorFlow Developers.** (n.d.). *https://www.tensorflow.org/*
   (For machine learning framework details.)
8. **He, K., Zhang, X., Ren, S., & Sun, J. (2016).** "Deep residual learning for image recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
   (For advanced CNN architectures that may apply to forgery detection.)
9. **ISO/IEC 27001.** (2013). *Information technology — Security techniques — Information security management systems — Requirements.*
   (For ethical and data privacy considerations.)
10. **Patil, D., & Sonawane, P. (2021).** "Survey on Document Forgery Detection Techniques." *International Research Journal of Engineering and Technology (IRJET)*, 8(3), 2021.