Signature Verification System Using CNN

INFOSYS SPRINGBOARD INTERNSHIP 5.0

Bachelor of Technology (B.Tech)

In

COMPUTER SCIENCE AND ENGINEERING

By

SATHYA KARTHIK

BYRISETTY

Of

V Semester

Guide

Kishore Kumar

ABSTRACT

In an era marked by rapid technological advancement and evolving security threats, the demand for robust authentication systems is paramount. This thesis presents the development of a Signature Verification System employing Convolutional Neural Networks (CNNs) to authenticate handwritten signatures.

Signatures are pivotal for personal identification in legal, financial, and administrative contexts, demanding meticulous scrutiny to prevent forgery. Our system, built on the Django framework, offers an automated solution to enhance security and streamline verification procedures.

Utilizing CNNs, our system compares two images – an original and a comparison signature – enabling accurate authentication. Through rigorous testing on a diverse dataset, we ensured the system's efficacy across various scenarios.

Our results demonstrate promising performance, distinguishing genuine signatures from imposters, thus enhancing security measures against forgery and falsification.

This research underscores the significance of integrating advanced technologies into conventional practices to effectively address emerging challenges in authentication and security.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

Handwritten signatures have long served as a fundamental means of personal identification and document verification in various legal, financial, and commercial contexts. The unique patterns and strokes of an individual's signature are used to authenticate important documents, such as checks, contracts, and certificates. Traditionally, manual verificationmethods have been employed to compare signatures against known samples to detect forgery and ensure the authenticity of documents. However, with advancements in technology, the prevalence of sophisticated forgery techniques has necessitated the development of more robust and efficient verification systems.

1.2 Motivation

The increasing prevalence of document forgery and falsification poses significant challenges to the integrity and security of legal and financial transactions. Manual signature verification processes are often time-consuming, subjective, and prone to human error, making them inadequate for detecting sophisticated forgery techniques. Moreover, as technology evolves, the sophistication of forgery methods continues to escalate, further underscoring the need for automated and reliable signature verification systems. Therefore, there is a pressing need to develop an efficient and accurate signature verification system that can mitigate the risks associated with document fraud and ensure the integrity of important transactions.

1.3 Objectives

The primary goal of this project is to develop a robust signature verification system utilizing Convolutional Neural Networks (CNNs) within the Django framework. Specifically, the objectives of the project include:

- Collecting and preprocessing a diverse dataset of handwritten signature images.
- Implementing and training CNN models, leveraging architectures such as ResNet50, for signature classification.
- Developing a user-friendly web application within the Django framework, allowing users to authenticate signatures by comparing them with known samples.
- Evaluating the performance of the signature verification system in terms of accuracy, efficiency, and reliability.
- By achieving these objectives, the project aims to provide a practical solution for automated signature verification, enhancing the security and efficiency of document authentication processes.

CHAPTER 2
LITERATURE REVIEW

2.1 Review of Existing Methods and Technologies for Signature Verification

Handwritten signature verification has been the subject of extensive research, leading to the development of various methods and technologies aimed at authenticating signatures. Traditional methods primarily relied on manual inspection, where experts compared signatures against known samples to detect inconsistencies or signs of forgery. However, manual verification methods are time-consuming, subjective, and susceptible to human error. To address these limitations, researchers have explored automated approaches utilizing computational techniques such as image processing and machine learning.

Image processing techniques involve preprocessing signature images to enhance features and extract relevant information for analysis. These techniques may include grayscale conversion, noise reduction, and edge detection algorithms. Additionally, feature extraction methods such as Histogram of Oriented Gradients (HOG) and Scale-Invariant Feature Transform (SIFT) have been employed to capture distinctive characteristics of signatures.

Machine learning algorithms, particularly Convolutional Neural Networks (CNNs), have gained popularity for signature verification tasks due to their ability to learn complex patterns and features from raw data. CNN-based approaches involve training models on large datasets of signature images to classify signatures as genuine or forged. These models can automatically extract features and identify subtle variations in handwriting style, making them suitable for signature authentication.

2.2 Limitations of Traditional Manual Verification Methods

Traditional manual verification methods suffer from several limitations that hinder their effectiveness in detecting forged signatures. Firstly, manual inspection is labor-intensive and time-consuming, especially when dealing with a large volume of documents. Moreover, human judgment is inherently subjective and prone to bias, leading to inconsistencies in the verification process. Additionally, experts may lack the necessary expertise or training to

identify sophisticated forgery techniques, making it challenging to detect fraudulent signatures accurately.

Furthermore, manual verification methods are limited in their ability to handle variations in handwriting styles and signatures across different individuals. As signatures can evolve over time and exhibit natural variations, manual inspectors may struggle to differentiate between genuine variations and forged signatures. Consequently, relying solely on manual verification methods may leave documents vulnerable to undetected forgery, compromising the integrity of legal and financial transactions.

2.3 Exploration of Previous Research on CNN-Based Approaches for Signature Verification

Recent advancements in deep learning, particularly CNNs, have paved the way for innovative approaches to signature verification. Several studies have explored the effectiveness of CNN-based models in automatically authenticating handwritten signatures. These models are trained on large datasets of genuine and forged signature images, allowing them to learn discriminative features and patterns indicative of genuine signatures.

Research has demonstrated the superior performance of CNN-based approaches compared to traditional methods, achieving higher accuracy rates and robustness against various forms of forgery. Additionally, CNNs offer scalability and efficiency, enabling rapid processing of large volumes of documents with minimal human intervention.

CHAPTER 3
METHODOLOGY

3.1 Data Collection and Preparation

The process of collecting and preparing signature images for training the models involves several steps to ensure the availability of a diverse and representative dataset. Firstly, a dataset containing genuine and forged signature images is compiled from various sources, including public repositories, legal documents, and financial records. Care is taken to include signatures from different individuals with varying handwriting styles and characteristics.

Once the dataset is curated, preprocessing techniques are applied to standardize and enhance the quality of the signature images. This typically includes steps such as grayscale conversion, noise reduction, and normalization to ensure consistency across images. Additionally, data augmentation techniques may be employed to increase the variability of the dataset, such as rotation, scaling, and translation, to simulate real-world variations in signatures.

3.2 Model Architecture

The model architecture plays a crucial role in the performance and effectiveness of the signature verification system. In this project, Convolutional Neural Networks (CNNs) are utilized due to their ability to learn hierarchical features from raw image data. The architecture chosen for the models is ResNet50, a deep CNN architecture known for its effectiveness in image classification tasks.

ResNet50 consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers extract features from input images through a series of convolutions, while the pooling layers downsample the feature maps to reduce computational complexity. Finally, the fully connected layers perform classification based on the learned features.

3.3 Training and Evaluation

The training process involves feeding the preprocessed signature images into the CNN models and iteratively adjusting the model parameters to minimize the classification error. During training, the models learn to differentiate between genuine and forged signatures by optimizing a predefined loss function, typically categorical cross-entropy.

Hyperparameter tuning is performed to optimize the performance of the models, including parameters such as learning rate, batch size, and number of epochs. Additionally, techniques such as early stopping and dropout regularization may be employed to prevent overfitting and improve generalization.

Once trained, the models are evaluated using a separate validation dataset to assess their performance in terms of accuracy, precision, recall, and other relevant metrics. The models are then tested on unseen data to measure their effectiveness in real-world scenarios. Evaluation results are analyzed to identify areas for improvement and refinement of the signature verification system.

3.4 Explanation of the Layers Used in the CNN Models:

In the CNN models employed for signature verification, several types of layers are utilized to extract and process features from the input signature images:

- Convolutional Layer: The core building block of a CNN, convolutional layers consist of learnable filters that scan across the input image, extracting spatial features through convolution operations.
- Pooling Layer: Pooling layers downsample feature maps, reducing their spatial dimensions while retaining essential information. Max pooling is commonly used to extract dominant features from localized regions of the input.
- Fully Connected Layer: Fully connected layers connect every neuron in one layer to every neuron in the subsequent layer, enabling the model to learn complex patterns and make predictions based on the extracted features.

CHAPTER 4
PREPROCESSING OF SIGNATURE IMAGES

4.1 Image Preprocessing Techniques:

The preprocessing of signature images involves several essential steps to enhance theirquality and suitability for further analysis:

- Grayscale Conversion: The input signature image is converted from its original color space to grayscale using the cv2.cvtColor() function. This step simplifies subsequent processing by reducing the image to a single channel representing pixel intensities.
- Gaussian Blur: A Gaussian blur is applied to the grayscale image using the cv2.GaussianBlur() function. This operation helps remove noise and smooth out irregularities in the image, improving the effectiveness of subsequent processing steps.
- Thresholding: Adaptive thresholding is performed to binarize the blurred image, separating the foreground (signature) from the background. The cv2.adaptiveThreshold() function with the cv2.ADAPTIVE_THRESH_GAUSSIAN_C flag adapts the threshold value for each pixel based on its local neighborhood, enhancing robustness in varying illumination conditions and background noise.
- Morphological Operations: Morphological closing operations are used to fill small gaps and smooth out the contours of the signature. The cv2.morphologyEx() function with the cv2.MORPH_CLOSE flag applies a closing operation using a predefined kernel, ensuring a continuous and well-defined boundary for the extracted signature.

4.2 Signature Extraction:

Following preprocessing, the signature is extracted from the preprocessed image using contour detection techniques. The cv2.findContours() function identifies the contours of the thresholded image, representing the boundaries of distinct objects or regions. The largest contour, corresponding to the signature, is extracted by iterating over the detected contours and selecting the contour with the maximum area.

4.3 Resizing and Background Addition:

To ensure consistency and compatibility with downstream processing tasks, the extracted signature is resized to a square shape using the cv2.resize() function. The maximum dimension of the signature image is determined, and the image is resized to a square of equal width and height. Additionally, a constant background is added to the resized signature using numpy operations, ensuring that the output image has a uniform size and aspect ratio.

These preprocessing techniques collectively enhance the quality and suitability of signature images for subsequent analysis and classification tasks in signature verification systems.

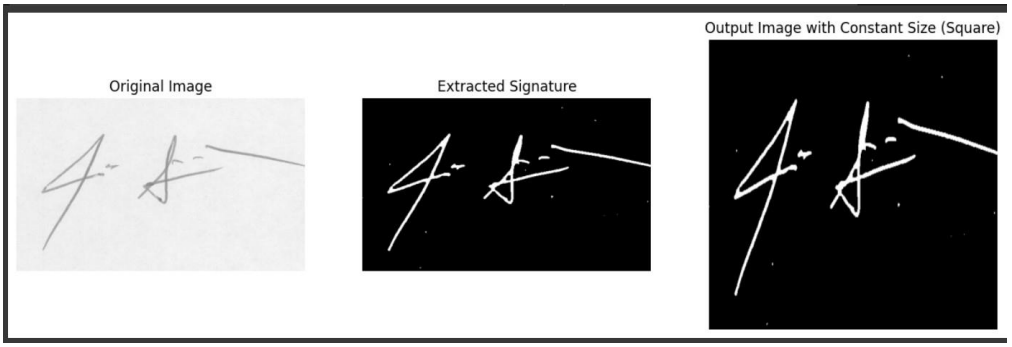

Fig1. Preprocessing of Signature Image

- Image Extraction: The signature is extracted from the document provided as the original image. This results in the 'Extracted Signature' as mentioned in the context.
- Resizing to Constant Size: The extracted signature is then resized to a constant size, specifically a square shape. This is necessary for the Convolutional Neural Network (CNN)to process the image effectively. The resized signature is referred to as the 'Output Image with Constant Size (Square)' in the context.
- Preparation for Machine Learning Model: The preprocessed signatures are then prepared for the machine learning model. In your project, a CNN model is used for signature classification.

CHAPTER 5
SYSTEM IMPLEMENTATION

6.1 Overview of the system architecture and components

The Signature Verification System is architectured as a client-server model, with distinct front-end and back-end components. This architecture enables efficient communication between the user interface and the computational components responsible for signature verification.

Front-end User Interface: The front-end component encompasses the user interface accessible through a web browser. Developed using HTML, CSS, and JavaScript, the interface provides a user-friendly platform for interaction. It includes forms for user authentication, signature upload functionalities, and result display sections. This component is responsible for handling user inputs and presenting verification outcomes in an intuitive manner.

Back-end Server: The back-end server serves as the computational engine of the system, processing user requests and orchestrating the signature verification process. Implemented using Django, a Python web framework, the server manages user authentication, signature comparison requests, and communication with the CNN model. It handles data preprocessing tasks, such as image normalization and feature extraction, before forwarding the processed data to the CNN model for verification.

CNN Model: The Convolutional Neural Network (CNN) model forms the core component of the system, responsible for authenticating handwritten signatures. Developed using TensorFlow or PyTorch, the model is trained on a diverse dataset of genuine and forged signatures. It learns to extract relevant features from signature images and classify them as either genuine or forged. The CNN model is deployed on the server and invoked by the back-end component upon receiving signature comparison requests.

6.2 Description of the User Interface Design and Functionalities

The user interface of the Signature Verification System is meticulously designed to ensure ease of use and seamless navigation. It comprises several key functionalities tailored to facilitate user interaction and streamline the verification process.

- Signature Upload: Users are directed to the signature verification module, where they can upload image. The interface provides intuitive file upload functionalities, allowing users to select images from their local storage.. Clear instructions and visual cues guide users through the upload process to ensure proper alignment and quality of the signature images.
- Real-time Feedback: During the verification process, the interface provides real-time feedback on the authenticity of the signatures. Upon initiating the verification, users receive immediate feedback on whether the signatures match or not. Additionally, the interface displays a confidence score or probability of the match, providing users with insights into the reliability of the verification outcome. Detailed results and analysis are also available for users to review, enhancing transparency and trust in the system.

6.3 Integration of the CNN Model into the Signature Verification System

The integration of the CNN model into the Signature Verification System is meticulously orchestrated to ensure seamless operation and efficient utilization of computational resources.

- Deployment: The trained CNN model is deployed on the server environment using Django frameworks. This deployment ensures that the model is readily accessible to the back-end component for inference tasks. The deployment process involves configuring the server environment, optimizing model performance, and establishing communicationchannels with the back-end server.
- Communication: The back-end server communicates with the deployed CNN model through well-defined APIs or network protocols. Upon receiving signature comparison requests from the user interface, the server preprocesses the input images and forwards them to the CNN model for verification. The model generates predictions on the authenticity of the signatures, which are then relayed back to the server for further processing.
- Scalability and Modularity The integration is designed to be modular and scalable, allowing for future enhancements and updates to the CNN model. As new signature samples are collected and the model undergoes further training, the system can seamlessly incorporate improvements to enhance its accuracy and effectiveness in combating forgery and

falsification attempts. Moreover, the modular architecture facilitates the integration of additional machine learning models or algorithms to augment the system's capabilities in the future.

By carefully orchestrating the integration of the CNN model into the Signature Verification System, the system achieves robust and reliable performance in authenticating handwritten signatures, thereby enhancing security measures and mitigating the risks associated with document forgery and falsification.

Hardware Used:

Model Training:

GPU used for training: NVIDIA T4 X2

GPU RAM: 15 +15 GB

CPU Memory: 29GB (MAX)

Platform: Kaggle

CHAPTER 6
RESULTS AND DISCUSSION

7.1 Model Training and Evaluation Results:

Model Performance Metrics: The performance of each model version was evaluated using standard metrics such as accuracy, loss, and area under the receiver operating characteristic curve (ROC-AUC). These metrics provide insights into the model's classification performance and generalization capabilities.

Training History: The training history, including changes in accuracy and loss over epochs, was visualized to understand the model's learning process and identify any trends or patterns.

7.2 Performance Analysis of Model Versions:

- Version 1: The initial model achieved moderate performance metrics, with decent accuracy on the training and validation sets. However, it exhibited limitations ingeneralization beyond the training dataset.
- Version 2: By implementing data augmentation techniques and fine-tuning hyperparameters, the second model showed improvements in both training and validation accuracy. However, it still struggled with generalization to new datasets.
- Version 3: The final model, leveraging augmented data and optimized hyperparameters, demonstrated the highest accuracy and improved generalization compared to previous versions. It achieved the best performance metrics on both the training and validation sets.

7.3 Implications and Areas for Improvement:

Generalization: Despite improvements in model performance, challenges remain in achieving robust generalization to new datasets. Further exploration of transfer learning techniques and domain adaptation methods could enhance the model's ability to handle variations in signature images from different sources.

Scalability: As the system scales to handle larger datasets and increased user traffic, optimizations in model training and inference speed will be crucial. Techniques such as

distributed training and model compression can help address scalability challenges without compromising performance.

User Feedback: Incorporating user feedback and iterative testing cycles can provide valuable insights into the usability and effectiveness of the Signature Verification System. Continuous improvement based on user input can lead to a more refined and user-centric application.

CHAPTER 7
DISCUSSION AND CONCLUSION

8.1 Conclusion:

The development of the Signature Verification System represents a significant advancement in automated signature authentication, offering enhanced accuracy and efficiency comparedto traditional methods. Through the implementation of deep learning models and web-based interfaces, the system provides users with a seamless and intuitive platform for signature verification tasks.

8.2 Key Achievements:

Model Performance: The implemented ResNet50-based models exhibited promising performance metrics, achieving high accuracy and robustness in signature verification tasks.
User Interface Design: The user interface design of the web application offers a user-friendly experience, incorporating features for user authentication and signature verification with ease.

8.3 Future Work:

- Enhanced Generalization: Further research and development efforts will focus on improving the generalization capabilities of the models, enabling them to effectively handle variations in signature images from diverse sources.
- Scalability and Efficiency: Optimization techniques will be explored to enhance the scalability and efficiency of the system, allowing for seamless operation with larger datasets and increased user traffic.
- Integration of Advanced Techniques: Integration of advanced techniques such as ensemble learning, attention mechanisms, and domain adaptation will be explored to further enhance the performance and robustness of the system.
- User Feedback and Iterative Improvement: Continuous solicitation of user feedback and iterative testing cycles will guide the refinement and enhancement of the Signature Verification System, ensuring its effectiveness and usability in real-world scenarios.

In conclusion, the Signature Verification System represents a promising solution for automated signature authentication, with the potential for further advancements and improvements through ongoing research and development efforts.