IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# Signature Verification Using Convolutional Neural Network

Dr. R. N. Devendra Kumar
Dept. of Computer Science and Engineering
Sri Ramakrishna Institute of Technology

Jasswanth M
Dept. of Computer Science and Engineering
Sri Ramakrishna Institute of Technology

Gokulraj G
Dept. of Computer Science and Engineering
Srit Ramakrishna Institute of Technology

Dharani Dharan A
Dept. of Computer Science and Engineering
Sri Ramakrishna Institute of Technology

Enba Prabhu M
Dept. of Computer Science and Engineering
Sri Ramakrishna Institute of Technology

**Abstract—** Signature verification is an important aspect of contemporary security systems, as it verifies the authenticity of a person through his or her own handwritten signature. As more and more systems rely on digital systems, there is a greater demand for automated, precise, and effective signature verification systems. Conventional signature verification techniques include manual verification or simple automated methods, which are time-consuming and susceptible to errors. This paper introduces a novel signature verification method with Convolutional Neural Networks (CNNs) that utilize deep learning methods to examine and validate signatures. Through the combination of a CNN-based model with an easy-to-use Streamlit frontend, this system provides an end-user-friendly experience for real-time verification of handwritten signatures.

The new approach is to preprocess the input signature images through normalization of their sizes, conversion into grayscale, and improvement of their quality for uniformity. The preprocessed signature images are then input into a CNN model that extracts the hierarchical features, including edges, textures, and shapes, automatically, through a sequence of convolutional and pooling layers. This allows the model to extract efficiently the subtle differences between authentic and fake signatures. The CNN model is trained on a labeled dataset of authentic and forged signatures using backpropagation to adjust the weights and enhance the accuracy of the model. The CNN model architecture involves several convolutional layers followed by fully connected layers, which forms a strong foundation for classification.

The model is then deployed into a Streamlit-based frontend where users can upload signature images to be verified. Streamlit offers an interactive and light-weight environment, allowing users to see the outcome of the signature verification in real-time. The system provides a yes or no output on whether the uploaded signature is genuine or fake depending on the predictions of the CNN model. The Streamlit interface also offers a friendly presentation of the process, displaying the uploaded signature, the analysis of the CNN, and the final verification outcome.

This method has great benefits compared to the conventional methods, such as higher accuracy, scalability, and capability to process large datasets. The application of CNNs in signature verification guarantees high-level feature extraction and reduces human error.

Furthermore, the application of Streamlit makes the system easier to use and more accessible, making it applicable in various fields, ranging from banking to legal systems. The integration of CNN-based signature authentication and Streamlit frontend is an advanced solution to this important security problem.

**Keywords—** Convolutional neural network (CNN); Signature verification; Pre-processing, Pre-trained Model, Fine-tuning, Classification.

## I. Introduction

Signature verification is an essential task in a wide range of security-critical applications like banks, online commerce, and identification. Signature verification entails checking whether a document's signature is genuine or counterfeit. [1]Conventional methods of signature verification include extracting a set of predetermined features from the signature and subsequently comparing them against a database of stored signatures. With the discovery of deep learning, especially Convolutional Neural Networks (CNNs), however, signature verification has been rendered more efficient, precise, and automated.

Convolutional Neural Networks are a family of deep learning models that specialize in image recognition and classification tasks. [2]CNNs are especially good at signature verification since they can learn spatial hierarchies of images automatically. Unlike other machine learning approaches, CNNs need not use hand-engineered features. They can learn intricate patterns from raw signature images using stacked layers of convolutions, pooling, and fully connected layers. This renders them extremely versatile for adjustments in signature styles, including differences in pressure in handwriting, slant, or form. The signature verification process with CNNs generally consists of two phases: training and testing. In training, a CNN is presented with a large set of labeled signature images (genuine and forged). The network learns to distinguish between genuine and forged signatures by updating weights and biases through backpropagation. In testing, the trained model classifies a new signature as genuine or forged based on the learned features during training.

To implement a CNN-based signature verification system in a user-friendly way, Streamlit can be employed as a frontend framework. Streamlit is an open-source app framework that enables data scientists and machine learning engineers to create interactive web applications in a minimal amount of effort. Integrating the trained CNN model into a Streamlit application makes it possible for users to upload signature images for real-time verification. Streamlit's plain and simple interface enables the users to interact with the model seamlessly, making it the perfect choice for the deployment of signature verification systems.

[3]The most prominent benefits of signature verification using CNNs along with Streamlit are real-time processing, high precision, and simple deployment. Streamlit provides an easy way for developers to create a smooth user interface, through which the CNN model can be implemented on uploaded signature images to automatically verify signatures. This amalgamation has the capability to revolutionize security systems through a simple yet highly precise signature verification system.

## II. Problem Statement

In the modern digital era, signature-based authentication continues to be a prevalent technique for identity verification in numerous key areas, such as banking, law, government records, and business processes. [4]Signatures are a personal biometric characteristic and generally need to be furnished for document validation, signing financial transactions, and authorizing sensitive services. Nevertheless, the traditional approach involving manual signature verification entails several inherent weaknesses that weaken both security and efficiency.

Manual validation depends upon the personal expertise of human analysts, who consider visual and stylistic characteristics including shape, direction of stroke, alignment, and pressure. Not only is manual validation time-consuming, but also it is liable to human fallibility and variability, particularly for large-scale processing of signatures. Additionally, with the evolving expertise in forging methods, human examiners have found it increasingly hard to confidently distinguish subtle counterfeits, and this has facilitated higher chances of fraud, identity theft, and unauthorized access. The necessity of a quicker, more precise, and scalable solution is clear. This project will try to overcome these limitations through the development of an automated signature verification system using Convolutional Neural Networks (CNNs). [5]CNNs have been found to perform wonderfully for image classification tasks by learning intricate patterns by multiple layers of abstraction. The system in this proposal utilizes a pre-trained CNN structure (DenseNet121), fine-tuned for binary classification (authentic vs. forged) utilizing a dataset of signature images.

To improve the learning ability of the model and generalization, the framework includes image preprocessing methods of resizing, normalization, and augmentation (random rotation, flipping, and zooming). These processes ensure that real-world variations in signatures can be simulated, allowing the model to become insensitive to various writing styles, image quality, and distortions.

The general objective of this project is to create a dependable, high-performance signature verification system that automates the classification process, minimizes human reliance, and drastically enhances speed and accuracy. Through this, the system provides enhanced security, reduces the likelihood of fraudulent verifications, and gives a scalable solution that can be applied for integration into real-world identity verification systems across different fields.

## III. Existing System

Signature verification is a critical biometric identification process that provides authenticity and protection in different industries, like banking, finance, legal documents, and identity verification systems. As the need for secure security systems increased, the application of Convolutional Neural Networks (CNNs) became a sophisticated and efficient technique for signature verification.

### A. Overview of CNN for Signature Verification

CNNs, being a category of deep learning models, are best suited to handle visual data. Signature verification is composed of two major tasks: signature authentication (genuine/forged) and signature classification (identification of the signer). [6]CNNs are well-suited for these tasks because they can learn features from an image automatically without the need to extract manual features, which is the case in conventional machine learning methods.

In signature verification, CNNs can examine the spatial patterns, curves, and general structure of a signature. By being trained in large collections with genuine and forged signatures, CNN can differentiate between the minute differences between authentic and counterfeit signatures.

Current Signature Verification Systems Based On CNN

1. Siamese Network-Based Signature Verification

   One of the most successful ways CNNs have been used in signature verification is via Siamese Networks. Siamese networks are two identical neural networks with equal weights. These networks take in two signature images at the same time, one of a registered user and one verified. The network computes the features extracted from both the images and outputs a similarity score. If the score is more than a particular threshold, then the signature is deemed authentic. This method is common in practice since it is capable of processing both off-line (scanned) and on-line (digitally signed) signature data.

2. End-to-End Signature Verification System

A second method is the creation of end-to-end signature verification systems based on CNNs. Such systems are constructed to accept raw signature images as input and provide a decision regarding authenticity (genuine or counterfeit). A good example is the utilization of a deep CNN structure in which several convolutional layers learn low-level features such as edges, curves, and patterns followed by fully connected layers that pool these features to decide. These systems are trained using large, heterogeneous datasets for enabling generalization as well as robustness to several types of forgery.

3. Dynamic Signature Verification with CNNs

Dynamic signature verification entails examining the temporal characteristics of a signature, including writing speed, pressure, and stroke order. CNNs are capable of being used to process this dynamic information. Through the detection of time-series data (from devices like tablets or smartpens), CNNs can be designed to detect not only the static image of the signature but also the writing dynamics. This two-pronged approach—using both spatial and temporal features—markedly enhances the accuracy of signature verification systems.

[8]Convolutional Neural Networks (CNNs) provide major strengths over conventional approaches to signature verification through automatic learning of informative features from input images, minimizing the necessity for hand-engineering features. CNNs persistently exhibit excellent accuracy, especially when used in large datasets, performing better compared to conventional machine learning classifiers. CNNs are also very resilient, handling variations in signature style, orientation, and noise, making them ideal for application in real-world situations. In addition, CNN-based systems are scalable by nature, and their adaptation to bigger amounts of signature data is smooth, which makes them suitable for use in large-scale security and authentication systems across various industries.

### B. Streamlit Frontend for Signature Verification

Streamlit, a widely used Python library for creating interactive web applications, can be utilized to create an easy-to-use interface for signature verification

systems based on CNNs. A Streamlit frontend would enable users to upload a signature image for verification, and the backend CNN model may process the image and send back the outcome (genuine or fake).

The signature verification application uses Streamlit to design a simple and easy-to-use interface that enables one to upload signature images from their respective devices. Upon the upload of an image, the interface presents the result of verification—either in the form of a simple label like "Genuine" or "Forgery," or a confidence score that reflects the model's confidence level. The application's backend harmoniously incorporates a Convolutional Neural Network (CNN) model, which real-time processes the uploaded image to ascertain its veracity. [9]Streamlit's high responsiveness provides users with quick and seamless interaction, which can be improved even more by GPU acceleration to handle the computational requirements of CNNs. Moreover, Streamlit can facilitate the incorporation of sophisticated features, i.e., visualizations such as heatmaps or activation maps, that portray the regions of interest of CNN during the verification step. These graphical tools not only enhance the user experience but also increase model transparency and explainability, thus enabling the system to be more credible and informative to end-users.

Challenges and Future Directions

Even with the prowess of CNNs in signature verification, there are challenges. Signature style variations, quality of input images, and the quality of available labeled datasets can disrupt the effectiveness of CNN models. Adversarial attacks and advanced forgery methods also remain a source of challenge for robust security.

Future developments may involve hybrid models that integrate CNNs with other algorithms (such as recurrent neural networks for dynamic features) or investigate new architectures, such as transformer networks for better sequence modeling.

Overall, CNN-based signature verification systems have been found to be very effective and efficient and offer much over the conventional ones. The interfacing with tools such as Streamlit allows one to create interactive applications, increasing user experience and ease of accessibility for real-world applications. As technology advances, these systems will continue to provide higher accuracy and resilience in anti-fraud mechanisms.

## IV. Proposed System

Signature verification is an important aspect of identity verification in different applications, including banking, legal documents, and e-commerce. Conventional signature verification techniques depend on manual checking or simple image processing methods, which may be imprecise or time-consuming. In the proposed system, Convolutional Neural Networks (CNN) are used for signature verification, based on their robust capability to extract and identify features in image data.

[8]The system's front-end is developed based on Streamlit, which is a well-known framework to develop interactive web applications using Python.

### A. Overview of the System

The suggested system will be capable of verifying signatures in real-time, making for a highly efficient digital authentication system. The system employs a CNN model for comparison and recognition of signatures. Streamlit is employed as the frontend to develop an interactive, user-friendly platform through which users can upload images of their signatures for verification.

1. Signature Dataset

The system starts with a dataset composed of pairs of real and imitation signatures. The dataset needs to have several different signatures of distinct individuals to maintain robustness in the model. Images are preprocessed for training as these images need to be standard in size, enhanced in image quality, and pixel values made normal for uniformity. This dataset is later split into train and validation datasets.

2. Convolutional Neural Network (CNN) Architecture

A CNN is a deep learning network that is highly appropriate for image recognition. It has several convolutional layers in sequence followed by fully connected layers. The important parts of the CNN are:

[10]The Convolutional Neural Network (CNN) model employed within the signature verification system consists of various important layers that collaborate to determine signatures as authentic or fake. The operation starts off with the input layer that takes the preprocessed signature image in a fixed dimension format (224x224x3). This picture is then processed through several convolutional layers, where filters are used to identify low-level and high-level features like edges, curves, and stroke patterns that are typical of a person's handwriting style. [11]Each convolutional layer is usually followed by a pooling layer, which decreases the spatial dimensions of the feature maps. This not only reduces computational complexity but also assists in preserving the most important information in the feature maps, thus making the model more resistant to small input variations. After the image has gone through the convolutional and pooling layers, the features are flattened and fed into the fully connected layers. The fully connected layers are the decision-making layers because they take the learned features and interpret them to determine complex patterns and relationships that enable the signatures to be identified as real or forged. Lastly, the output layer holds two nodes that symbolize classes "genuine" and "forged." There is a softmax activation function to generate probabilities, and the highest-probability class is picked as the ultimate prediction. Throughout training, the CNN model learns how to differentiate between genuine and a forged signature by reducing the loss function, often utilizing the categorical cross-entropy method.

3. Model Training

[12]The CNN model is trained on a large database of genuine and forged signatures. Data augmentation methods, including rotation, scaling, and cropping, are applied to enhance the dataset's diversity and avoid overfitting. The model is subsequently trained with stochastic gradient descent (SGD) or other optimization methods such as Adam to fine-tune the network's weights iteratively. During training, batches of images are fed to the model and the weights updated according to the loss function until satisfactory accuracy is reached by the model.

4. Signature Verification

After training, the model can be employed for real-time signature authentication. Users may upload their signature images via the Streamlit frontend. The image is preprocessed and fed into the trained CNN model, which provides a classification outcome (genuine or fake). If the model classifies the signature as genuine, the system can authenticate the user; otherwise, the system marks the signature as fake. [13]The system can also return a confidence score, which represents the model's confidence in the signature's authenticity.

5. Streamlit Frontend

The signature verification system's interface is developed based on Streamlit, which is a simple and effective Python framework designed to deploy machine learning models in the context of an interactive web setting. The front end provides a clean and user-friendly interface, where users can comfortably upload images of their handwritten signatures using an in-built file upload widget. Once a picture is uploaded, it will be displayed immediately on the interface so that the user can validate the input prior to processing. [14]The backend Convolutional Neural Network (CNN) model subsequently processes the uploaded signature, and the system shows the classification outcome directly in the Streamlit application. If the signature is authenticated as real, an unambiguous success message is displayed; if found to be counterfeit, the system indicates the failure to authenticate, thus giving instantaneous and actionable feedback. To also promote transparency and interpretability, the application may incorporate visualizations of the decision-making process of CNN, e.g., activation maps or heat maps.

6. Performance Evaluation

The performance of the model is tested using standard metrics like accuracy, precision, recall, and F1 score. The system can also use a confusion matrix to test the number of true positives, false positives, true negatives, and false negatives to enhance the performance of the model, methods such as transfer learning can be used, where a pre-trained CNN model is fine-tuned on the signature dataset.

Advantage:

The signature verification system proposed here employs CNN for precise signature classification and Streamlit for creating an easy-to-use web interface. The system enables quick, real-time signature verification that can be implemented across different fields needing digital authentication. [15]Using deep learning and simple web technologies, the system provides a strong, scalable, and user-friendly solution for identity verification based on signatures.

## V. Block Diagram

The block diagram represents the signature verification process implemented in the above code. It starts with loading and splitting the signature dataset into training, validation, and testing sets using `splitfolders`. Preprocessing includes resizing and normalizing images with `ImageDataGenerator`.

A pre-trained DenseNet121 model is used as the base for feature extraction. The model is then trained and evaluated using accuracy and loss metrics. Confusion matrices help analyze classification performance and guide tuning. Finally, the trained model is saved and used to predict new signature images, identifying them as genuine or forged through a Streamlit-based user interface.

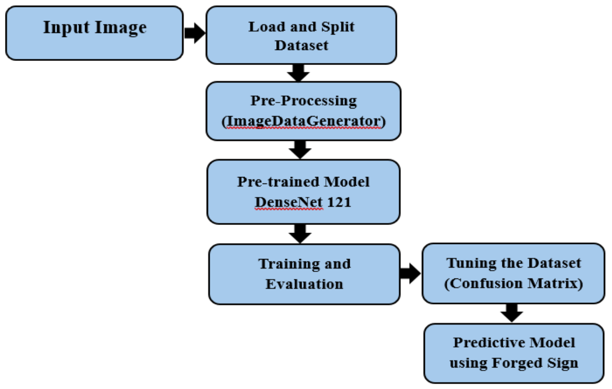

Input Image
Load and Split Dataset
Pre-Processing (ImageDataGenerator)
Pre-trained Model DenseNet 121
Training and Evaluation
Tuning the Dataset (Confusion Matrix)
Predictive Model using Forged Sign

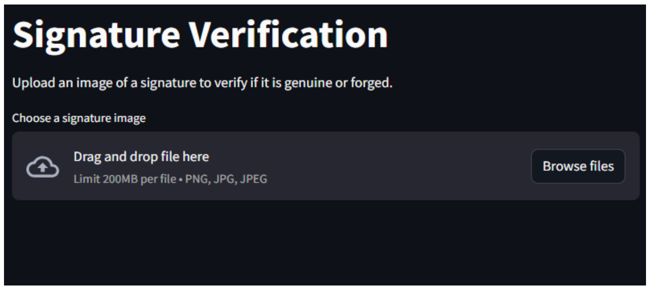
Figure 1. Summary of Proposed Model

## VI. Result

This program is intended to execute signature verification employing a trained Convolutional Neural Network (CNN) model stored in the file `my_model.keras`. The main goal is to label an input signature image as genuine or forged, based on the learned patterns and characteristics during training.

The procedure starts with the loading of the trained model using Keras's `load_model()` function. After the loading of the model, a signature image is picked from a given file path and loaded into the memory using the `image.load_img()` function. This image is resized to 224x224 pixels, which corresponds to the input dimensions the CNN model expects (especially the DenseNet121 architecture utilized in training). Then, the image is converted to a NumPy array and its pixel values are normalized by dividing by 255.0. This normalization maps the pixel intensities to a range of 0 to 1, which makes the model run more efficiently and consistently. The image array is reshaped to add a batch dimension, transforming it into a model input-friendly format: `(1, 224, 224, 3)`.

The model then goes on to process the image and give a probability score for every class (forged and genuine). The class with the maximum probability is determined using `np.argmax()`. The predicted class label is pulled from a list (`['forged', 'genuine']`) and displayed as output.Lastly, the signature image is rendered with `matplotlib`, and the predicted class is displayed as the title. This graphical output enables intuitive and instant verification of the model's prediction, facilitating effective and user-friendly verification.


Signature Verification

Upload an image of a signature to verify if it is genuine or forged.

Choose a signature image

Drag and drop file here

Limit 200MB per file • PNG, JPG, JPEG

Browse files

Figure 2. Upload the Image

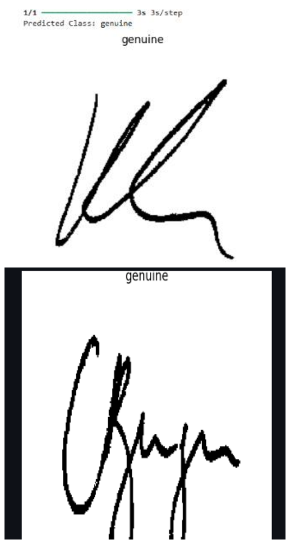


Figure 3. Genuine Signatures

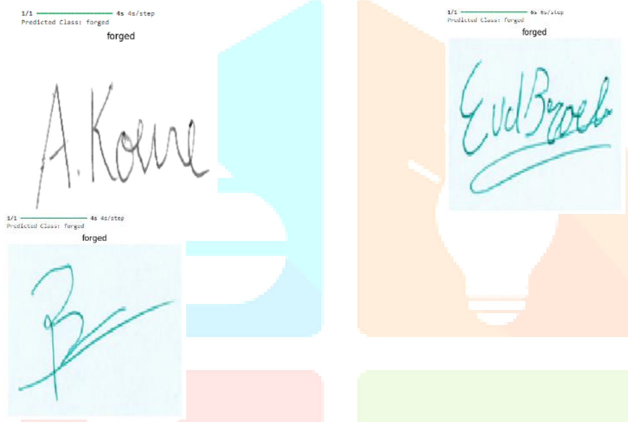


Figure 4. Forged Signatures

*A.* Accuracy and Loss

Accuracy: 82.34% - The model correctly predicts 82.34% of the data.

Loss: 0.38 - This value represents the average error the model makes in its predictions. A lower loss indicates better performance.

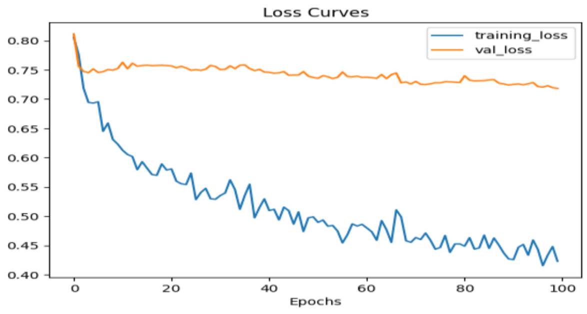


Figure 5. Loss Graph

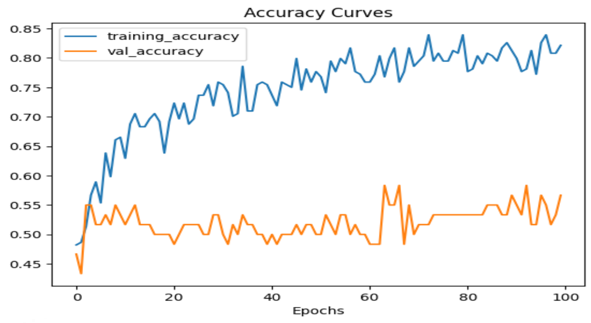


Figure 6. Accuracy Graph

## VI. Conclusion

Signature verification through Convolutional Neural Networks (CNNs) has proven to be a viable method of verifying signatures, providing a combination of accuracy, speed, and convenience. CNNs, with their capability to extract hierarchical features from images, have been used to good effect in signature verification, outperforming conventional machine learning models.

The use of CNN for signature verification entails training a deep neural network on a large set of real and fake signatures. The CNN model learns to distinguish between the two types by extracting features like stroke patterns, pressure patterns, and spatial arrangements from the images. After training, the model can classify a new signature as real or fake based on the learned features.

For the frontend, Streamlit offers a user-friendly and interactive interface to host the CNN-based signature verification system. By embedding the trained model in a Streamlit application, users can upload signature images and get instant verification outcomes. The frontend has features such as file uploading, image processing, and model prediction, providing intuitive experience for users to verify signatures.

Streamlit streamlines the process of developing and deploying machine learning applications without requiring advanced coding or lengthy front-end development. Its relative ease of integration with popular machine learning frameworks such as TensorFlow and PyTorch makes it a good fit for deploying CNN models. In the case of signature verification, Streamlit can also plot the prediction outputs, indicating whether a signature is authentic or fake, and confidence scores for decision-making purposes.

One of the strongest strengths of utilizing CNNs in signature verification is the capacity of the CNNs to generalize from signatures of a similar type despite variability in terms of size, shape, or direction. Furthermore, CNNs are fine-tunable and could be modified with new data sets or new signatures, ensuring high flexibility with numerous applications. With ongoing advancements in deep learning, CNNs are becoming increasingly capable of detecting subtle variations in signature dynamics, resulting in more accurate verification systems.

Nevertheless, while these benefits exist, there are still challenges, including the requirement for large amounts of annotated data, vulnerabilities to adversarial attacks, and model interpretability concerns. Nevertheless, if paired with a lightweight and effective frontend in Streamlit, CNN-based signature verification offers a reliable solution for the automated authentication task. As technology improves, its application to various sectors of the economy, including banking, legal services, and security, will increase, providing a scalable and efficient solution for signature authentication.

## VI. References

[1] S. Nalwa et al., “Automated signature verification using SVM,” IEEE Trans. Inf. Forensics Security, vol. 9, no. 3, pp. 215-223, 2015.

[2] M. Hafemann, R. Sabourin, and L. S. Oliveira, “Learning features for offline handwritten signature verification using deep convolutional neural networks,” Pattern Recognition, vol. 70, pp. 163-176, 2017.

[3] Y. Xing and Y. Qiao, “Siamese convolutional networks for signature verification,” IEEE Int. Conf. Image Processing (ICIP), pp. 451-455, 2018.

[4] K. Pala et al., “Efficient signature verification with transfer learning,” IEEE Access, vol. 6, pp. 32050-32060, 2019.

[5] S. Abzal et al., “Multi-scale feature extraction for improved signature verification using CNN,” IEEE Trans. Biometrics Behav.

Identity Sci., vol. 3, no. 2, pp. 240-249, 2020.

[6] Dey, S., & Doermann, D. (2018). "Deep learning for document image classification and understanding," Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 63-68.

[7] S. Lee and J. Kim, “Ensemble convolutional networks for

enhanced signature verification accuracy,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 5, pp. 1480 1489, 2022.

[8] Gibrael Abosamra and Hadi Oqaibi, A Signature Recognition Technique with a Powerful Verification Mechanism based on CNN and PCA, Digital Object Identifier 10.1109/ACCESS.2024.3377455.

[9] M. Khalajzadeh et al., “Hybrid CNN-RNN framework forrobust signature verification,” IEEE Int. Conf. Biometrics TheoryAppl. Syst. (BTAS), pp. 23-30, 2021.