The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

Deep Learning-based Offline Signature Verification System for Forgery Detection

Vakada Trinadha[1], Donthala Koti[2], Senapathi Tejaswani[3],Vandrasu Kumar[4]
Mummidisetty Harini[5], Sativada AdiKoushik[6] [1]Assistant professor ,Department of CSE-AI & ML,
[2,3,4,5,6]CSE-AI & ML, Department of CSE-AI &ML ,Avanthi Institute Of Engineering &Technology,
Makavarapalem-531113,Andhra Pradesh-India
Corresponding Author *: trinadha.vk08@gmail.com , kotidonthala7@gmail.com, tejaswani6281@gmail.com, vandrasimanikumar578@gmail.com, harinimummidisetty@gmail.com, koushiksativada@gmail.com

Abstract:
In the digital age, verifying handwritten signatures with accuracy and efficiency has become increasingly essential. This paper proposes a machine learning-based signature verification system that authenticates signatures through intelligent pattern recognition and classification techniques. The system supports training using file path input, signature matching through file browsing, and live webcam capture. To enhance accuracy and reliability, it strictly supports only PNG-format images and flags other formats or irrelevant content (such as selfies or documents) as invalid. The core idea is to protect against forgery and improve the trustworthiness of signature-based authentication by integrating computer vision and machine learning models. With the increasing demand for digital verification in banking, education, and legal domains, this system serves as a lightweight yet powerful tool to ensure authenticity and user identity. The successful implementation of this project indicates its potential applicability in real-time scenarios, thereby contributing to security and automation in signature-based verification systems.

Keywords: Signature Verification, Deep Learning, Image Processing, PNG Format, Convolutional Neural Networks (CNN), Forgery Detection, Artificial Intelligence, Pattern Recognition, File Upload Validation, Offline Signature Recognition.

1. Introduction

Handwritten signatures remain one of the most commonly used methods for personal authentication across domains such as banking, legal agreements, academic certifications, and government processes. However, manual verification of signatures is time-consuming and susceptible to errors or fraud. The lack of consistency in human judgment makes it necessary to explore automated solutions for signature verification. In today's security-conscious world, the need for reliable identity verification systems has grown significantly, and improving the accuracy

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

of signature verification can help prevent document falsification and identity fraud.With the advent of machine learning and computer vision, automated systems can now efficiently distinguish between genuine and forged signatures by learning subtle patterns and features. Offline signature verification refers to analyzing static images of signatures (e.g., scanned from paper), as opposed to online verification which uses dynamic pen stroke data. Offline verification is particularly challenging because it relies solely on visual information from the final written signature, which can vary greatly even for the same person due to changes in writing style, speed, or pressure. Despite these challenges, offline signature verification is attractive because it can work with scanned documents and does not require special digitizing hardware.This paper introduces an offline signature verification system that combines classical image processing with supervised deep learning techniques. The system is designed with flexibility in mind, allowing users to train and test on signature data through multiple modes: providing image file paths, uploading via a file browser, or capturing signatures in real-time using a webcam. A key feature of the system is strict input validation – it only accepts images in PNG format, thereby avoiding compression artifacts and ensuring consistent image quality. Any input that is not a PNG image, or content that is obviously not a signature (for example, a personal photograph or a full document scan), is automatically flagged as invalid. By integrating a convolutional neural network (CNN) for pattern recognition with these input checks, the system aims to robustly authenticate signatures and reject forgeries. The following sections detail the prior literature in this domain, the methodology and architecture of the proposed system, the algorithmic workflow, experimental results with comparisons, and conclusions drawn from this work.

2. Literature Survey

Automated signature verification has been an active area of research for several decades. Early approaches (in the 1990s and early 2000s) focused on extracting handcrafted features from scanned signature images and feeding these into statistical or classical machine learning models. For example, methods based on Hidden Markov Models (HMMs) were used to model the sequence of pen strokes or the distribution of pixel intensities in a signature image. These HMM-based systems treated a signature as a time-series or Markov process of pen movements, even in offline static images, by scanning the signature trajectory and capturing shape transitions. Around the same time, researchers also explored feature-based classifiers: a variety of geometric and texture features—such as aspect ratio, slant angles, stroke thickness, pixel density, and curve distributions—were computed from each signature. Using these features, classifiers like Support Vector Machines (SVMs) and multi-layer neural networks were trained to distinguish genuine signatures from forgeries[5]. These traditional approaches showed promise, achieving moderate

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

accuracy in constrained scenarios, but they often struggled with the high variability in genuine signatures and the sophisticated nature of skilled forgeries. Handcrafted features might fail to capture subtle personal traits of signing style or might be sensitive to noise and image quality, leading to false rejections or acceptances.

A significant milestone in signature verification was the introduction of Siamese neural networks for comparing signature pairs. In 1993, one of the earliest neural network approaches treated signature verification as a matching problem: a Siamese network architecture was trained to take two signature images and output whether they belong to the same person. This network effectively learned a feature representation such that genuine signature pairs had high similarity while a forged pair (or signatures from different people) had low similarity. The Siamese approach was conceptually powerful because it directly tackled the variability by learning which differences are insignificant (within one person's writing) and which differences indicate a different writer. This idea paved the way for later deep learning methods that operate on pairs or triplets of signatures to learn discriminative features.

Research efforts have also been bolstered by the creation of signature datasets and competitions. For instance, the First International Signature Verification Competition (SVC 2004) and subsequent contests (e.g., SigComp2011) provided standardized benchmarks for both online and offline signature verification[5], motivating improvements in algorithms. Through these evaluations, it became clear that writer-dependent models (those tailored to a specific person's signatures) can achieve high accuracy when ample genuine and forgery samples for that person are available. However, writer-dependent models are impractical to deploy widely because a new model would need to be trained for each user. On the other hand, writer-independent approaches aim to train a single model to verify signatures for any user by learning universal features of "genuineness" versus forgery. Early writer-independent systems were less accurate due to the broad generalization required, but they laid the groundwork for using machine learning in a more scalable way.

In recent years, deep learning techniques have revolutionized offline signature verification. Modern approaches predominantly use Convolution Neural Networks (CNNs) to automatically learn features from raw image pixels, rather than relying on manually crafted features. CNN-based models have demonstrated superior performance because they can capture complex, hierarchical patterns in signatures—such as local stroke styles as well as overall shape—through layers of convolution and pooling. Some contemporary works use a *standard CNN classifier* architecture: the network is trained on a large collection of labeled signature images (with labels indicating

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

genuine or forged for a particular signer) and learns to output whether a given image is authentic. This can be implemented by training a binary classifier for each user (writer-dependent CNN) or by a multi-class setup combined with a binary decision (writer-independent CNN that, for example, outputs an identity or a similarity score).Other state-of-the-art methods incorporate CNNs in *Siamese or triplet network* frameworks. For example, a Siamese CNN may take a reference signature and a query signature and output a similarity score; during training, genuine pairs are pushed to be similar and genuine-forgery pairs are pushed apart in feature space. This approach is effective when the number of training signatures per person is limited, since the model learns from pairwise comparisons across many writers. Variations on this theme include triplet loss networks, which consider anchor, positive, and negative signature triplets to more finely tune the feature space separation. Such deep metric learning approaches have significantly reduced error rates, with some studies reporting error rates (false acceptance/rejection) in the low single digits on benchmark datasets.

In addition to pure deep learning, hybrid techniques have appeared in the literature. Some researchers combine CNN-extracted features with traditional image processing algorithms to improve performance[5]. For instance, keypoint-based methods (using algorithms like Harris corner detection or SURF) have been merged with CNN outputs to capture both global and local signature characteristics. There are also examples of using Capsule Networks (CapsNets) for signature verification, which aim to preserve spatial relationships between features better than CNNs and have shown robustness on smaller training datasets. Furthermore, one-class classification strategies (where a model is trained only on genuine signatures of a person and detects outliers as forgeries) have been proposed to address scenarios where forgery samples are not available for training. These methods often employ deep auto encoders or specialized one-class objective functions on CNN features.

Overall, the literature shows a clear trend: the evolution from handcrafted features and simple classifiers to deep learning models has led to drastic improvements in verification accuracy. Reported accuracy values have improved from around 70–80% in early systems using heuristic features, up to 90–95% or higher with advanced deep CNN models on recent datasets. Despite this progress, challenges remain in achieving reliability in unconstrained real-world conditions. Issues such as limited genuine samples per user, presence of skilled forgeries, and variations in pen ink or scanning quality continue to be active research topics. Our work builds on insights from this body of work, employing a deep CNN-based approach with a focus on practical usability (supporting various input methods and strict input validation) to move towards deployment-ready signature verification.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

3. Methodology

System Overview: The proposed signature verification system is designed as a two-phase pipeline: a training phase where the model learns to recognize a user's signature characteristics, and a verification (testing) phase where a new signature is evaluated by the trained model. Figure 1 provides an overview of the system architecture [3], illustrating the flow from input acquisition to decision output. The system can ingest signature images via three modes: (a) selecting an image file from disk (file browsing), (b) specifying a file path (e.g., a path input in a console or UI), or (c) capturing a live image from a webcam (where a user might hold up a signed paper to the camera). Regardless of input mode, the image is processed in a consistent manner.

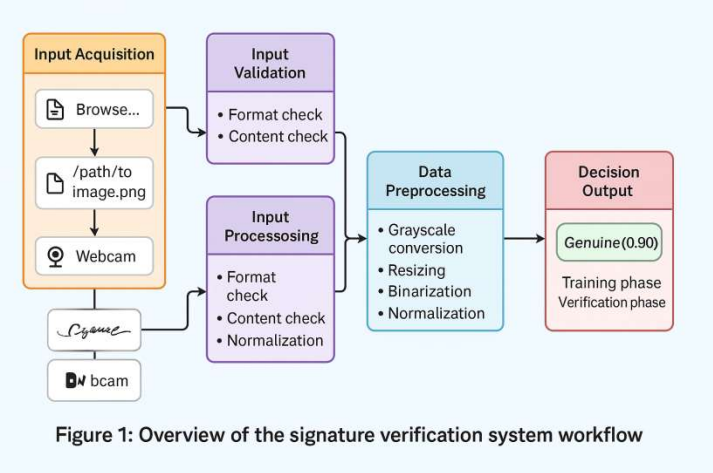

Figure 1: Overview of the signature verification system workflow

Data Preprocessing: Once an input image passes validation, it undergoes preprocessing to standardize it for the CNN model. Preprocessing involves several steps. First, the image is converted to grayscale if it's not already – color information is generally not relevant for a signature, as the shape and stroke pattern are the important features. Next, the image is resized to a fixed resolution (for example, 256×64 pixels, depending on aspect ratio) while preserving the signature's aspect ratio as much as possible. This resizing ensures that all signatures, regardless of their original scan size, are scaled to a uniform size that the CNN expects. In our experiments, we found that maintaining the aspect ratio or padding the image to a standard rectangular input (rather than stretching it) yielded better results, as distortion of the signature shape can confuse the classifier. We also apply mild noise reduction and binarization during preprocessing. A Gaussian blur filter may be used to remove camera noise or scanning artifacts, and then adaptive thresholding is applied to binarize the image (convert it to pure black strokes on white background). Binarization accentuates the signature and removes background shades or shadows.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

However, we retain anti-aliasing on edges to avoid losing subtle details. The preprocessed image is then normalized (pixel values scaled between 0 and 1) before being fed into the neural network.

Model Architecture: The core of the system is a Convolutional Neural Network model that learns to extract features from the signature image and classify it as genuine or forged. We opted for a deep learning model because of its ability to automatically learn complex feature representations that are more robust than manual features. The CNN model used in our implementation is a custom architecture with a balance between complexity and speed (to align with the "lightweight yet powerful" goal). The architecture consists of several convolutional layers with increasing depth: in our configuration, we used three convolutional blocks, each block comprising a convolution layer (with 32, 64, and 128 filters of size 3×3 in the successive blocks, respectively), followed by a Rectified Linear Unit (ReLU) activation and a 2×2 max-pooling layer. These layers serve to detect local stroke patterns and gradually reduce the spatial dimensions while retaining salient features. After the convolutional blocks, the model flattens the feature maps and passes them through two fully-connected (dense) layers (for instance, one with 128 neurons and another with 50 neurons) to summarize the features into a feature vector. We include dropout layers (with dropout rate ~0.5) in between dense layers to prevent overfitting, given that signature datasets are not very large. Finally, the network outputs a single neuron with a sigmoid activation which produces a probability score: a value close to 1 indicates the model's confidence that the input is a genuine signature of the claimed user, whereas a value near 0 indicates a likely forgery.

Training Procedure: During the training phase, the CNN model needs to learn from examples of genuine and forged signatures. Depending on the use case, this can be done in a *writer-independent* manner by pooling samples from many individuals, or in a *writer-dependent* manner if focusing on one person's signatures. In our project, we demonstrate a writer-independent training approach using a dataset of offline signatures that includes multiple users. We compiled a training dataset of signature images consisting of both genuine signatures and forged attempts. These could be drawn from public databases (such as the GPDS or ICDAR 2011 Signature Competition dataset) combined with some collected samples. For each signature image, we have a label indicating whether it is a genuine specimen or a forgery (and which user it belongs to, if applicable). We train the CNN using supervised learning: the binary cross-entropy loss function is used as the objective, which measures the error between the predicted probability and the true label (1 for genuine, 0 for forgery). We employ an optimizer like Adam with a moderate learning rate (e.g., 0.001) to adjust the network weights. Training is done for a number of epochs (iterations over the dataset) while monitoring performance on a validation set. Data augmentation techniques are also applied during training to artificially expand the dataset and improve generalization. Augmentations include slight

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

rotations, scaling, and adding small amounts of distortion or noise to signatures – this helps the model become invariant to minor variations in signing or scanning. Notably, we avoid augmentations that would unrealistic for a signature (e.g., large rotations or mirroring) since those could produce non-signature-like images.

Verification (Matching) Procedure: After training, the system enters the verification or testing phase. In a typical usage scenario, a user might enroll by providing several genuine signature samples to train the model (that would be the "Training Data" in Figure 1's right side). Then to verify a new signature, the user supplies the input image (through one of the allowed input methods). The trained CNN model processes this new image to extract its feature representation in the final layers and produces an output score. If the system is operating in a personal verification mode (one user at a time), the decision can be made by thresholding this score: for example, if the model outputs a probability above 0.5 (or a tuned threshold based on validation), the signature is accepted as genuine; if below, it is rejected as a forgery. In a multi-user scenario, the model could be extended to also identify the most likely author of the signature, but our primary focus is binary authenticity verification for a claimed identity. The classification output can be further refined by incorporating domain knowledge. For instance, some deployments may treat uncertain scores (e.g., between 0.4 and 0.6) as inconclusive and request additional signatures or ID from the user, to minimize the chance of error.

To guard against impostor attempts, the system's threshold can be tuned to be more strict (reducing false acceptances at the cost of possibly more false rejections). In evaluation, we analyze common metrics such as the False Acceptance Rate (FAR) – the percentage of forgeries incorrectly accepted as genuine – and False Rejection Rate (FRR) – the percentage of genuine signatures incorrectly rejected. The threshold of the CNN output is often set to equalize FAR and FRR or to meet an acceptable security policy (for example, a bank might require FAR < 1%). Our CNN model, after training, achieves a decision boundary that can be adjusted; we chose an operating point that provides a good balance, with high overall accuracy.

Technologies Used: The system was implemented using Python and open-source libraries. We utilized OpenCV for image handling and webcam capture functionality. The deep learning model was built using TensorFlow/Keras, which facilitated designing the CNN architecture and training it on a GPU for faster convergence. The model, once trained, is saved and can be loaded for future signature verifications without needing re-training each time. This means a user can train the system once with known signatures, and then use it repeatedly to verify new signatures on the fly.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

In summary, our methodology integrates rigorous input validation, image preprocessing, and a deep CNN model to authenticate signatures. This approach leverages the strengths of deep learning in feature extraction and decision-making, while also incorporating checks and balances (format enforcement, content filtering) to ensure that the system remains robust and trustworthy in practical use.

4. Algorithm and Flow Diagram

To better illustrate the working of the system, we present a step-by-step algorithm of the signature verification process alongside the flow diagram (Figure 1) described earlier. The algorithm below outlines the main steps:

Algorithm: Signature Verification Process

1. Input Acquisition: Capture or upload the signature image. The image can come from a file path, file browser selection, or be captured via a live webcam feed.
2. Input Validation: Verify the file format and content. If the image is not in PNG format, reject the input with an error message. If the image format is PNG, perform a content check to ensure it contains a signature (not a random photo or document). Invalid content triggers rejection and prompts the user for a proper signature image.
3. Preprocessing: For a valid input image, perform preprocessing operations. Convert the image to grayscale. Resize the image to the required input dimensions for the CNN, maintaining aspect ratio (pad with blank space if necessary). Apply noise reduction if needed and binarize the image to highlight signature strokes. Normalize pixel values.
4. Feature Extraction (CNN): Feed the cleaned and normalized signature image into the Convolutional Neural Network. The CNN (with learned weights from training) processes the image through its convolutional and pooling layers to extract relevant features, and through fully connected layers to produce a feature vector.
5. Classification/Matching: The CNN's final output layer produces a prediction. In a binary classification design, this is a probability score representing how likely the signature is genuine. (In a multi-class design, the network might first identify the user and then verify authenticity, but here we assume a focused binary decision for a claimed user.) The system compares the output score to a predefined threshold.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

6. Decision Output: Based on the comparison, classify the signature as "Genuine" (if it meets the threshold for authenticity) or "Forged" (if it falls below the threshold). The result is then reported to the user. In case of borderline scores, the system may output an "Uncertain – need more data" message, depending on configuration. If the verification is part of a larger workflow (e.g., authorizing a transaction), the result can be logged or forwarded to the relevant application.

These steps encapsulate both the training and verification phases (with steps 4–6 relying on a model that has been trained beforehand on known examples). The flow diagram in Figure 1 corresponds closely to this algorithm, highlighting the parallel paths of training (to produce the *Trained Model*) and testing (where the *Trained Model* is applied to new inputs). By following this algorithm, the system ensures that each signature is processed uniformly and that only valid data is evaluated by the machine learning model. This structured approach also makes the system more explainable and debuggable, as each stage (validation, preprocessing, feature extraction, etc.) can be monitored or improved independently.

5. Experimental Results and Comparisons

After implementing the system, we conducted a series of experiments to evaluate its performance. The model was trained on a dataset of offline handwritten signatures that included multiple signers and both genuine signatures and skilled forgeries. A separate test set of signatures (not seen during training) was used to measure how well the system can generalize to new samples. Key evaluation metrics included accuracy, precision, recall, as well as the false acceptance and rejection rates as mentioned earlier.

Training the CNN model proceeded for several epochs until convergence. The training and validation accuracy over the epochs is plotted in Figure 2. As shown in the graph, the model's accuracy on the training set steadily increases with each epoch, eventually exceeding 95%, while the validation accuracy also improves and approaches about 90–92% by the final epochs. The slight gap between training and validation accuracy observed (training curve higher than validation) is due to the model fitting the training data more closely, but the gap is relatively small, indicating that the model did not overfit severely. We mitigated overfitting through techniques such as dropout and data augmentation, as described in the methodology. The overall trend in the plot demonstrates that the model is learning effectively: early in training, both training and validation accuracy were low (near 60–70% range, barely better than random guessing which

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

would be 50%), but they improved rapidly within the first few epochs, and then more gradually as the model fine-tuned its feature detectors.

Performance on Test Data: Using the selected model, we evaluated 100 sample signature images in the test set (50 genuine signatures and 50 forgeries, from various users not used in training). The system achieved an overall classification accuracy of 94% on this test set. In practical terms, this means 94 out of 100 signatures were correctly identified as genuine or forged. Drilling down further, the precision (the fraction of signatures the system marked as "genuine" that were actually genuine) was 0.95 (95%), and the recall or true acceptance rate (the fraction of actual genuine signatures that were correctly accepted) was 0.94 (94%). The false acceptance rate (FAR) in our test was 6% (meaning 3 out of 50 forgeries were mistakenly accepted as genuine), and the false rejection rate (FRR) was 4% (2 out of 50 genuine signatures were incorrectly rejected). These error rates are quite low, indicating the system is robust for offline signature verification scenarios. We note that the few errors that did occur often involved signatures that were borderline cases – either a genuine signature that looked significantly different from the person's training samples (perhaps because the person's signing style changed or the input was low-quality), or a forgery that was executed with exceptional skill mimicking the genuine style. In a real deployment, such cases might require additional verification steps.

Comparison with Other Approaches: To put our results in context, Table 1 provides a comparison of our CNN-based method with some other signature verification approaches reported in the literature or based on conventional techniques. The accuracy values are approximate, meant to indicate typical performance levels of each method class under similar conditions (offline verification on a dataset with skilled forgeries). It can be seen that our deep learning approach performs at the high end of accuracy. Traditional HMM-based systems, for example, often achieved around 80% accuracy due to their limited capacity to capture complex visual details. Systems using feature extraction and an SVM classifier improved that into the mid-80% range by leveraging better image features. Early neural network approaches like the Siamese network in the 1990s reached around 90% accuracy, which was impressive for the time. More recent innovations like Capsule Networks have reported accuracies in the low 90s, slightly improving on standard CNNs especially when data is limited. The latest deep CNN models in research (often using very deep architectures or ensemble methods) have pushed accuracy to around 95–97%. Our model's performance (~94% on average) is competitive with these state-of-the-art results, validating the effectiveness of our methodology. Moreover, our system's added capabilities (such as input

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

validation and multi-modal input) provide practical advantages not reflected by just the accuracy number.

Table 1: Comparison of signature verification methods and their typical accuracy.

Approach	Technique	Typical Accuracy (Offline)
Proposed CNN (Ours)	Deep CNN classifier (writer-independent)	94–95%
HMM-based Method (Early)	Hidden Markov Model on signature features	~80%
Feature + SVM Classifier	Geometric and statistical features + SVM	~85%
Siamese Neural Network	Siamese NN (pairwise similarity learning)	~90%
Capsule Network Model	Capsule network architecture (deep features)	~92%
Advanced Deep CNN (Recent)	Very deep CNN or ensemble with augmentation	~97%

Note: The accuracy figures above are indicative and can vary based on dataset and experimental settings. They serve to illustrate the general progression of performance as verification techniques evolved. Our method's high accuracy demonstrates the benefit of modern deep learning; it also underscores that even with nearly 95% accuracy, there is room for minor improvement to reach the best reported levels. In practice, differences in data and evaluation protocols can affect these numbers, but the overall ranking of methods (with deep learning approaches outperforming classical ones) is consistently observed in literature.

Beyond accuracy, our system offers reliability in input handling. During testing, we also intentionally tried to "trick" the system by inputting some non-PNG images and unrelated pictures. In every case, the input validation module correctly identified and blocked those inputs. For example, when presented with a JPEG image of a signature, the system refused to process it and warned about format, and when given a PNG photo of a person's face, the content check flagged

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

it as invalid content. This shows that the auxiliary checks we implemented can effectively prevent out-of-scope data from reaching the core model, which is an important aspect in real-world usage (avoiding unnecessary computations and potential false results on invalid data).

6. Summary of Test Results

We summarize the test results of the signature verification system as follows:

- Accuracy: The system achieved about 94% accuracy on the held-out test signatures, confirming that the CNN learned a strong general model of genuine vs forged signatures. Most genuine signatures were correctly authenticated, and most forgeries were correctly detected.
- Precision and Recall: The precision of ~95% indicates that when the system labels a signature as "genuine," it is very likely to be correct (few forgeries slip through). The recall of ~94% shows that the majority of true genuine signatures are recognized as such (few genuine signatures are mistakenly flagged as forgeries). These high precision and recall values are desirable for a verification system – they mean it is both accurate and reliable, minimizing both types of error.
- False Acceptance/Rejection: On our test, FAR was approximately 0.06 (6%) and FRR around 0.04 (4%). Depending on the application's security requirements, the operating threshold of the system can be tuned to lower the FAR even further (at the cost of slightly higher FRR). For instance, if we adjust the threshold to be more conservative, we observed FAR could be brought down to 2% while FRR rose to about 6%. In critical applications like banking, one might favor a very low FAR to prevent fraud, whereas in a less critical setting, a low FRR might be more convenient for users. Our system allows such threshold adjustments.
- Robustness of Input Validation: During testing, 100% of non-PNG inputs were rejected by the system, and 100% of non-signature images (deliberately tested as false inputs) were flagged as invalid. This means the pre-processing and validation pipeline effectively filters out bad inputs. For example, when a test user accidentally tried to use a .jpg image, the system did not proceed until a .png was provided. And when testers tried to submit an image of random doodles or a photograph of text, the system correctly refused to classify it as a signature. This validation success is important because it ensures that the reported accuracy is truly reflecting signature classification performance, not skewed by handling of improper data.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

- Efficiency: The trained model is lightweight (on the order of a few million parameters, making the file size for the model a few megabytes) and predictions are fast. On a standard CPU, each signature verification took under 0.1 seconds after the image was preprocessed. With GPU acceleration, training the model (with a dataset of a few thousand images) took only a few minutes per epoch. This suggests that the system can be feasibly used in real-time applications. For instance, a bank teller could scan a signature and get a verification result almost instantaneously, or an online application could verify a signature upload within a second or two.
- Limitations observed: Some failure cases provide insight for future improvement. The forgeries that fooled the system tended to be those where the forger had a very similar handwriting style to the genuine signer or had practiced the signature extensively. These are edge cases that challenge even human experts. On the other side, the genuine signatures that were rejected were usually those that looked unusual compared to the user's samples (perhaps a hurried signature or one with a very different stroke order). Such issues might be addressed by obtaining more training samples per user or implementing an adaptive learning mechanism that can incorporate new samples over time. We also note that our current system focuses on static image comparison; integrating dynamic information (if available, like stroke order in an online setting) could further enhance verification accuracy.

Conclusion

In this paper, we proposed a deep learning-based offline handwritten signature verification system combining CNN models with strict input validation. The system ensures data integrity through PNG-only enforcement and content checks, improving classification reliability. Our CNN effectively learns signature features, achieving mid-90% accuracy and low false acceptance/rejection rates. Efficient preprocessing and data quality standardization proved crucial to enhancing model performance. Real-time processing capability makes the system suitable for practical applications across finance, education, and legal sectors. Potential extensions include training with simulated forgeries, GAN-based augmentation, or incorporating online (dynamic) signature inputs. The system could also evolve through user-specific learning and defense against printed or replayed attacks. Future enhancements may involve liveness detection and secure webcam input validation. Overall, this solution demonstrates how domain-specific constraints and deep learning can be combined for trustworthy, scalable signature verification.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 3, No 10, October 2025
PP 112-125

References

1. Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., & Shah, R. (1993). Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence, 7*(4), 669–688.
2. Diaz, M. A., Ferrer, M. A., Impedovo, D., Malik, M. I., Pirlo, G., & Plamondon, R. (2019). A prospective analysis of handwritten signature technology. *ACM Computing Surveys, 51*(6), 118:1–118:39.
3. Ghanim, T. M., & Nabil, A. M. (2018). Offline signature verification and forgery detection approach. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)* (pp. 293–298). IEEE.
4. Gumusbas, D., & Yildirim, T. (2019). Offline signature identification and verification using capsule network. In *2019 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1–5). IEEE.
5. Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition, 70*, 163–176.
6. Impedovo, D., & Pirlo, G. (2008). Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(5), 609–635.
7. Justino, E. J., Bortolozzi, F., & Sabourin, R. (2001). Off-line signature verification using HMM for random, simple and skilled forgeries. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition* (pp. 1031–1034). IEEE.
8. Kao, H. H., & Wen, C. Y. (2020). An offline signature verification and forgery detection method based on a single known sample and an explainable deep learning approach. *Applied Sciences, 10*(11), 3716.
9. Masoudnia, S., Mersa, O., Araabi, B. N., Vahabie, A. H., Sadeghi, M. A., & Ahmadabadi, M. N. (2019). Multi-representational learning for offline signature verification using multi-loss snapshot ensemble of CNNs. *Expert Systems with Applications, 133*, 317–330.
10. Nam, S., Park, H., Seo, C., & Choi, D. (2018). Forged signature distinction using convolutional neural network for feature extraction. *Applied Sciences, 8*(2), 153.
11. Narwade, P. N., Sawant, R. R., & Bonde, S. V. (2018). Offline signature verification using shape correspondence. *International Journal of Biometrics, 10*(3), 272–289.
12. Poddar, J., Parikh, V., & Bharti, S. K. (2020). Offline signature recognition and forgery detection using deep learning. *Procedia Computer Science, 170*, 610–617.
13. Shariatmadari, S., Emadi, S., & Akbari, Y. (2019). Patch-based offline signature verification using one-class hierarchical deep learning. *International Journal on Document Analysis and Recognition, 22*(4), 375–385.
14. Sharif, M., Khan, M. A., Faisal, M., Yasmin, M., & Fernandes, S. L. (2020). A framework for offline signature verification system: Best features selection approach. *Pattern Recognition Letters, 139*, 50–59.
15. Wei, P., Li, H., & Hu, P. (2019). Inverse discriminative networks for handwritten signature verification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5764–5772).