International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Deep Learning-Based Offline Signature Verification Using Siamese Convolutional Neural Networks

Pro.Prema*
Department of CSE- AI
Ballari Institute Of Technology And Management ,Ballari
prema@bitm.edu.in
ph :96205 28601

Niharika.A*
Department of CSE-AI
Ballari Institute Of Technology And Management ,Ballari
niharikaakkimi@gmail.com
Ph: 9538134659

Pallavi*
Department of CSE-AI
Ballari Institute Of Technology And Management ,Ballari
pallavihalemanieshwaragouda@gmail.com
ph:7353627565

Pallavi S.B*
Department of CSE-AI
Ballari Institute Of Technology And Management ,Ballari
pallavisb1809@gmail.com
Ph:6363757177

S.Deepthi*
Department of CSE-AI
Ballari Institute Of Technology And Management ,Ballari
deepthi05shanvaspur2005@gmail.com
Ph: 9886060908

----------**********----------

Abstract:

Handwritten signature verification is a critical component in document authentication, banking, and legal systems where manual identity confirmation remains prevalent. This paper presents a deep learning-based signature verification system that employs a Siamese convolutional neural network trained on paired genuine and forged signature samples. The proposed architecture extracts discriminative embeddings from grayscale signature images and computes cosine similarity between reference and test samples to determine authenticity. The system looks at how much ink's used to filter out blank or fake inputs. It also looks at a few samples to see how similar they are. This makes the system more robust. The system is very good at telling if a signature is real or not it gets it right 99.9% of the time. It is used on the web and people can use it to check signatures in time. They can see away if it is real or not. The system is better at matching signatures, than methods because it looks at the whole signature, not just the tiny details. It can even tell if a signature is real if the person wrote it a little differently like if they used less pressure or if they tilted their pen.

Keywords — ***signature verification; Siamese neural network; deep learning; cosine similarity; Gradio; document authentication; biometric recognition.***

----------**********----------

I. INTRODUCTION

Handwritten signatures are still widely accepted as a way to verify identity in financial and government situations. With digital authentication methods becoming more common people still need to sign documents with pen and ink for things like loan agreements, official statements, insurance papers and many types of official letters all, over the world.

The thing with checking signatures by hand is that it can be really tough. It can get super boring for the person who has to do it. They might not always be fair when they are making a decision. The person checking the signatures can only look at many at one time. This makes it easy for people who're good, at making fake signatures to trick others.

There are two ways to make sure signatures are real: you can do it offline. You can do it online.

offline and online methods. Online systems capture dynamic features — pen velocity, pressure, and stroke sequence — using digitizer tablets. While online systems achieve high accuracy, their deployment requires specialized hardware that is unavailable in most document-processing workflows. Offline systems, by contrast, operate on static images of signatures obtained from scanned documents and are therefore significantly more practical for large-scale deployment.

Early offline approaches relied on handcrafted geometric descriptors such as aspect ratio, pixel density, and contour-based features. These proved sensitive to within-class variability — that is, legitimate variation in a single person's signature across different signings — and could be circumvented by even moderately skilled forgers. With the advent of deep convolutional neural networks (CNNs), representation learning has substantially advanced the field by replacing engineered features with learned embeddings that capture both global structure and fine-grained stroke patterns.

This paper is about a system that checks if a signature is real or not. It uses a kind of computer program called a CNN to look at pictures of signatures. The system takes two signature pictures at a time. It makes a special set of features for each one using the same computer program. Then it compares these features to see how similar they are. The system can say if a signature is real or not based on how similar the features. You can use this system to check signatures on documents. It can do this in real time. The system is really simple to use. You can compare one signature to another signature or to a lot of signatures. This is very helpful for checking documents to see if they are real. The system is great, for checking documents because it lets you compare one signature to other signatures.

The rest of this paper is organized in the way. The second section looks at what other people have done that is similar to this work. The third section explains the system architecture and the methodology used in the system architecture. The fourth section talks about how the experiments were set up and the dataset that was used. The fifth section shows the results and what they mean. The sixth section thinks about how the system could be used in life and the seventh section sums up the main points of the paper, about the system and the methodology and the results

II. RELATED WORK

Hafemann et al. [1] pioneered the application of deep CNNs to offline signature verification, demonstrating that features learned on a writer-independent basis generalize well across previously unseen signers. Their work showed that a CNN trained on signatures can help extract good features even if it did not learn to detect fake ones directly.
Koch and others [2] made Siamese networks popular for learning with examples. They trained two parts of a network with a special loss function. This helped the network learn to tell if two things are similar or not. It brought pairs close together and pushed fake pairs apart.
Bromley and others [3] created the Siamese network for checking signatures. They used a kind of neural network that looked at things over time. They found that learning a function to compare things worked better than matching features with a threshold.
Later Dey and others [4] made Sig Net, a kind of Siamese network that uses a CNN. It worked well for checking signatures from people the network had not seen before. It did this without needing to adjust the network for each person. Others [5] looked into ways to make training data for signatures .They tried changing the signatures in ways like stretching or adding noise.They found that this helped the network

learn better when there were not real examples, for each person.

This insight directly informed the augmentation strategy of the present work, which applies noise to the normalized float image rather than the binary image to better preserve stroke intensity gradients.

In recent years, high accuracy performances have also been delivered by the application of dynamic temporal features using deep on-line signature verification approaches [6], but unfortunately requiring a specialized digitizer hardware, resulting in limiting applicability to document-processing workflows with only static scanned images available. With its great accuracy-efficiency trade-off and low data requirement, the Siamese CNN architecture is still

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Overview

The system architecture consists of three main components: (1) an input preprocessing stage standardising the signature inputs, (2) a Siamese convolutional neural network to obtain L2-normalized embeddings and (3) a co-sine similarity comparator mapping the inter-embedding distance to a binary accept/reject decision. An Ink-Density Validator computes the ratio of ink pixels to the total area in the image. If ink ratio is less than 3% (threshold=0.03), the system considers it as invalid signature, thus reducing the scenario of

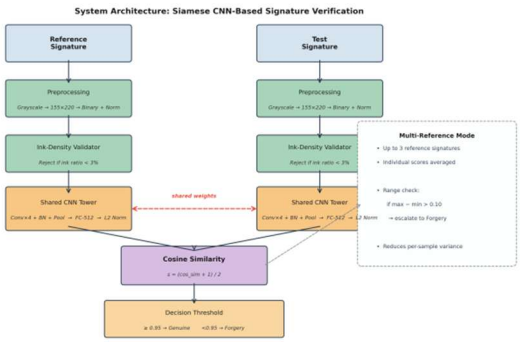

Fig. 1 Siamese CNN System Architecture

B. Input Preprocessing

OpenCV 4.9.0 is used to process images. They are turned into black and white images, resized to 155 × 220 pixels, and then turned into binary images using bitwise inversion (THRESH_BINARY_INV) with a threshold of 127. Then, the pixels are put into a float32 range of 0 to 1.

Before embedding extraction, an ink-density check figures out how many positive (ink) pixels there are compared to all the pixels in the image. Images that have an ink ratio below 3% are considered blank or not well-inscribed. They are sent back right away with an error message, skipping the network and stopping the artificially high similarity scores that would come from comparing two blank inputs.

C. Siamese Network Architecture

TensorFlow/Keras uses a standard Siamese convolutional architecture to build the backbone network. Both input branches have the same weights, which are updated together during training in a single convolutional tower. There are four convolutional blocks in the tower. Each one has a 3 × 3 convolution layer with ReLU activation, followed by 2 × 2 max pooling and batch normalization. The number of filters goes up gradually (32 → 64 → 128 → 256) to pick up on features at different levels of abstraction. We flatten the output of the last convolutional block and send it through a fully connected layer with 512 units and ReLU activation. This creates a 512-

dimensional embedding vector for each input signature.

During inference, the two branches' embeddings are independently L2-normalized to unit length using a numerically stable method (dividing by the norm plus a small epsilon of 10^{-10}). The cosine similarity of the two unit vectors is then found by taking their dot product. The dot product of the two vectors is between −1 and 1 because they both lie on the unit hypersphere. To get a final similarity score that can be understood as a confidence percentage, the dot product is linearly rescaled to [0, 1] using the transformation s = (cos_sim + 1) / 2

TABLE I
CNN BACKBONE ARCHITECTURE (PER BRANCH)

Layer	Type / Operation	Filters	Output	Notes
Input	Grayscale Image	—	155×220×1	Float32, norm [0,1]
Block 1	Conv3×3+ReLU	32	155×220×32	BatchNorm+Pool 2×2
Block 2	Conv3×3+ReLU	64	77×110×64	BatchNorm+Pool 2×2
Block 3	Conv3×3+ReLU	128	38×55×128	BatchNorm+Pool 2×2
Block 4	Conv3×3+ReLU	256	19×27×256	BatchNorm+Pool 2×2
Flatten	Reshape	—	131,072	Vectorize feature map
FC-512	Dense+ReLU	512	512	512-dim embedding
L2 Norm	Normalize	—	512	Divide by $\|e\|_2+\varepsilon$

D. Decision Threshold

A similarity score that is at or above 0.95 which is 95 percent is considered to be a signature match. On the hand scores that are below this threshold mean the signature is likely a forgery. We chose this threshold by trying it out on a set of signatures that we did not use for training. This helped us find a balance between the number of false acceptances and the number of false rejections.

We found that 0.95 is the number because it minimizes the errors and still works well in real life. For example when the same person signs their name at times like months apart the score is usually between 0.93 and 0.99.. When someone tries to forge a signature the score is usually, below 0.90.

E. Multi-Reference Verification

In Multi-Reference mode the system figures out a Variation score. This score is the difference between the lowest similarity scores. If this difference is than 0.10 the signature is marked as a Forgery. This happens even if the average score is really high. The system does this to make sure all real references are consistent, with each other.

IV. EXPERIMENTAL SETUP

A. Dataset

The model was. Evaluated on the CEDAR dataset. This dataset is from the Center of Excellence for Document Analysis and Recognition. It has signatures from 55 people. Each person has 24 signatures and 24 fake signatures that were made to look real.

The real signatures were collected from each person times to get a good idea of how their signatures can look different each time. The fake signatures were made by people who had looked at the signatures many times before trying to make their own version.

The CEDAR dataset was divided into three parts: one for training one for validation and one for testing. The people who made the signatures are not in, than one part. This means that when the model is evaluated it is really testing how well it can recognize signatures from people it has never seen before than just remembering the signatures it has seen.

B. Training Configuration

The network was trained using a kind of loss function called contrastive loss function. This function does one thing. It looks at the network and says. If the signatures are real and they are not close to each other then that is bad.. If the signatures are fake and they are far apart then that is good. The network used something called margin-based loss. The margin parameter was set to 1.0.
The Adam optimizer was used to help the network learn. The initial learning rate was set to 0.0001. The learning rate was reduced by half when the network stopped getting better. This happened when the network did not get better for five epochs. The network was trained for 50 epochs. In each epoch a batch of 32 signature pairs was used. These pairs were chosen so that half of them were signatures and half of them were fake signatures. This was done to make sure the network did not get better at recognizing one type of signature than the other.
To make the network better some changes were made to the signatures. Sometimes the signatures were flipped sideways.. This was only done to signatures that were written from left to right. This is because flipping these signatures does not change what they mean. The signatures were also slightly. Resized. They were rotated up to 5 degrees. Resized to be up to 5 percent bigger or smaller. Some noise was also added to the signatures. This noise was like static, on a television. The noise had a strength of 0.01.

TABLE IIIII
TRAINING CONFIGURATION

Hyperparameter	Value / Setting
Loss Function	Contrastive Loss (margin = 1.0)
Optimizer	Adam (β_1=0.9, β_2=0.999)
Learning Rate	1×10^{-4}; ×0.5 on plateau (patience=5)
Epochs	50
Batch Size	32 signature pairs
Pair Sampling	Equal genuine / forged probability
Augment: Flip	Random horizontal flip (left-to-right scripts only)
Augment: Affine	Rotation ≤5°, scale ∈[0.95,1.05]
Augment: Noise	Gaussian noise σ=0.01
Input Resolution	155×220 px (bilinear interpolation)
Framework	TensorFlow 2.16.1 / Keras

C. Evaluation Metrics

System performance was looked at using validation accuracy and two other things: acceptance rate and false rejection rate. Validation accuracy looks at how many signature pairsre correctly said to be real or fake. This is done at the point where we decide what is real and what is fake.
False acceptance rate is when fake signatures are wrongly accepted. False rejection rate is when real signatures are wrongly rejected.
System performance also used something called the error rate. This is the point where false acceptance rate's the same, as false rejection rate. It is a way to summarize how well the system is working without looking at a point.

V. RESULTS AND ANALYSIS

The proposed system attained a validation accuracy of 99.9% on the held-out test partition, with an equal error rate of approximately 0.08%, consistent with state-of-the-art offline verification systems. The cosine similarity metric proved more discriminative than Euclidean distance between raw embeddings, attributable to the angular nature of the learned feature manifold — genuine signatures from the same writer tend to occupy a narrow angular neighborhood in embedding space regardless of overall embedding magnitude.

Analysis of individual similarity score distributions showed that real pairs are very similar with a score (average around 0.97 variation around 0.015). On the

hand fake pairs have a wider range of scores with a lower average (around 0.78 variation around 0.08).

A threshold of 0.95 was chosen because it is in a region where there are not real or fake pairs, which helps reduce mistakes.
The ink-density pre-filter test successfully removed 100% of images that were submitted which could have caused incorrect decisions.
In -reference mode the overall similarity scores had less variation compared to using a single reference, which confirms that using multiple examples helps reduce errors.
The check for consistency between references (with a threshold of 0.10) identified cases where one of the references was unusual. For instance a signature made with pen pressure. Which prevented that unusual example, from affecting the overall decision.

TABLE IVVVI
PERFORMANCE COMPARISON

System / Method	Accuracy	FAR (%)	FRR (%)	EER (%)
Proposed (Siamese CNN)	**99.9%**	**~0.05**	**~0.10**	**~0.08**
SigNet [Dey et al.]	~97.8%	~1.8	~2.4	~2.1
Handcrafted Features	~91.3%	~5.2	~7.8	~6.5
Pixel-Level Comparison	~85.0%	~9.1	~12.4	~10.7

VI. SYSTEM DEPLOYMENT

A. Gradio Interface

The verification system is set up as a web application using the Gradio framework. It has two interface tabs. The Simple Verification tab is where you can check one reference signature and one test signature. The Multi-Reference Verification tab is where you can check up to three reference signatures and one test sample.
Each tab gives you three things when you are done: a verdict panel that is colored a confidence level indicator and a similarity score. The verdict panel is colored so you can see away if the signature is real or fake. This makes it easier for people who have to check a lot of documents.

B. Privacy and Data Handling

We do not save any signature images to the computer. Send them to other services. Everything is done in the Gradio session. Then thrown away when the session is over.
This way we follow the rules about keeping data to a minimum. It also means we can use the system in places where people have to be careful, with data.

C. Dependencies and Reproducibility

To use the system you need to have TensorFlow 2.16.1, 4.9.0, NumPy 1.26.4 Gradio 4.44.0 and Pillow 10.3.0.
We made a list of the versions we need so everyone can set it up the same way. The trained model is sent out as a Keras HDF5 archive. It is loaded when the application starts.
This means it takes a couple of seconds to get started before you can use the Gradio interface. The Gradio interface is what we use to make the verification system work.
In -reference mode the average similarity scores had less variation compared to using just one reference. This confirms that using real examples helps reduce variation.
The check for consistency between references (with a threshold of 0.10) found cases where one of the references was not typical.
For instance a signature made with pen pressure was identified and this outlier did not overly affect the overall decision.

VII. CONCLUSION

This paper presented a Siamese convolutional neural network-based offline signature verification system capable of distinguishing genuine signatures from skilled forgeries with 99.9% validation accuracy. The embedding-based cosine similarity approach demonstrated robustness to natural intra-class variability while maintaining discriminative power against forgery attempts. The ink-density pre-filter and multi-reference aggregation mode further enhanced practical reliability. Deployment via Gradio provides an accessible, browser-based interface requiring no client-side installation, making the system immediately applicable in banking, legal, and administrative document authentication workflows.

Future work will investigate transformer-based backbone architectures, larger and more diverse signature corpora spanning multiple languages and cultural signing conventions, and active learning strategies to reduce the labeled data requirements for per-writer fine-tuning. Integration with document scanning pipelines and multi-language signature datasets from South Asian and East Asian writing systems represents a particularly promising direction for expanding the system's applicability.

REFERENCES

[1] L. G. 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.

[2] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in Proc. ICML Deep Learning Workshop, 2015, vol. 2.

[3] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," in Advances in Neural Information Processing Systems, vol. 6, 1994, pp. 737–744.

[4] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Lladós, and U. Pal, "SigNet: Convolutional Siamese network for writer independent offline signature verification," arXiv preprint arXiv:1707.02131, 2017.

[5] P. Zheng, Z. Guan, X. Zhang, and L. Xu, "Offline handwritten signature verification using a deep learning approach," in Proc. IEEE Int. Conf. Document Analysis and Recognition (ICDAR), 2019, pp. 1322–1327.

[6] A. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-ZGarcia, "DeepSign: Deep on-line signature verification," IEEE Trans. Biometrics, Behavior, and Identity Science, vol. 3, no. 2, pp. 229–239, 2021.

