

Offline Signature Verification Using

Machine Learning:

ANASS ESSAFI

University of Abdelmalek Essaadi, National School of Applied Science Al-Hoceima

Department of Mathematics and Computer Science

Abstract :

In environments that rely on offline signatures for authentication, verifying the authenticity of signatures is crucial to maintain system security. This paper explores a machine learning approach to offline signature verification, focusing on distinguishing between genuine and forged signatures using a dataset of handwritten samples. The study emphasizes the challenges of ensuring reliability in signature verification systems and discusses the methods and results achieved. This research contributes to advance automated solutions for offline signature verification, highlighting its potential to strengthen security in critical applications.

• Introduction :

In biometric authentication, signature verification plays a crucial role in identity validation across various domains, including legal and financial sectors (in banks for example)...

Signature can be either online or offline, this study focuses specifically on the offline signature analysis.

Traditional manual signature comparison relies heavily on expert human judgment, which is a time consuming and prone to errors, and with the increasing sophistication of forgery techniques, there is a pressing need for some automated solutions and systems.

Recent advancement in machine learning and computer vision have opened new avenues for developing some accurate and reliable signature verification systems. However existing researches often struggle with the complex and nuanced characteristics that differentiate genuine signatures from forgeries, not to mention the lack

of large, diverse datasets.

In this research, we aim to improve this field by leveraging machine learning techniques to transform signature verification from a complex, error-prone process into a more reliable and precise identification method. Our methodology addresses the critical challenges of signature authentication by learning robust feature representations that capture the intricate nuances of individual signature styles. Through our model, we aim to enhance the ability to differentiate between authentic and fraudulent signatures, and that will strengthen security in critical applications.

• Related works :

In recent years Signature verification has been extensively studied in the field of biometric authentication with different approaches developed to address the problem.

These methods aim to leverage the interpretability of handcrafted features while enhancing performance through

robust classifiers, achieving high accuracy on benchmark datasets like GPDS and CEDAR. However, these methods typically require large labeled datasets and significant computational resources, limiting their applicability in resource-constrained environments. Despite these advancements, challenges such as handling skilled forgeries, achieving generalization

across diverse datasets, and developing lightweight models remain significant. Our study addresses these gaps by adopting an approach with an optimized preprocessing pipeline. By combining geometric and texture-based feature extraction with effective scaling methods, our work aims to provide an efficient and interpretable solution for offline signature verification.

- **Methodology :**

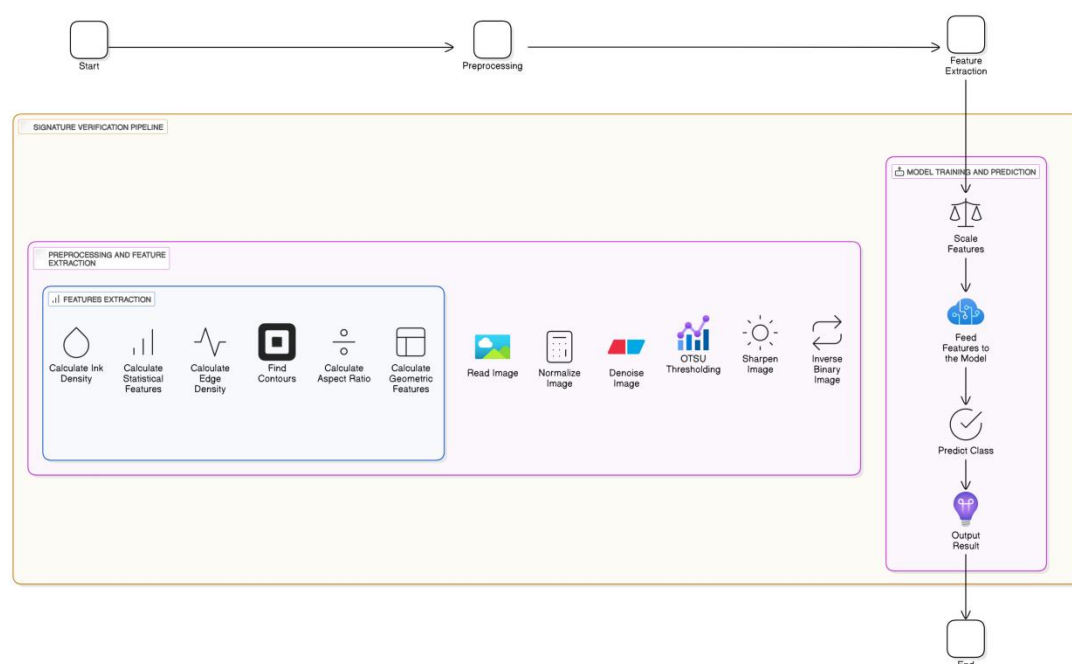


Figure 1 : Signature verification pipeline

This project methodology involves a systematic approach, starting with the collection of signature datasets that contains both genuine and forged specimen, ensuring data quality through comprehensive preprocessing and feature extractions techniques designed to transform raw signature images into significant numbers to the models. This approach begins firstly by labeling images into 0 for forgeries and 1 for

genuine, then we applied advanced image preprocessing, including Gaussian denoising, OTSU thresholding and image sharpening, to enhance signature clarity and remove noise.

Feature extraction employs a multi-dimensional methodology by combining statistical features such as ink density, mean intensity, skewness, and kurtosis with geometric characteristics like contour analysis, aspect ratio,

additionally, advanced texture analysis is applied using Local Phase Quantization (LPQ).

And after extracting those features, we passed them into machine learning models like SVM, Logistic regression... to classify the signature as either genuine or forged, with the dataset split into training and testing samples using a 80-20 stratified split.

This standard train-test partitioning approach ensures that 80% of the data is used for model training, while 20% is reserved for testing the model's performance.

The stratified sampling technique ensures that the distribution of genuine and forged signatures remains consistent between both sets, reducing the possibility of class imbalance.

- **Data :**

For this signature verification project we have gathered some well-known datasets to make the model more used to different types of signatures with good quality.

The first one is the CEDAR dataset that contains a collection of scanned handwritten signatures.

55 writers who came from different cultural backgrounds, each of them provided 24 original signatures, in total 1,320 original images, additionally we had also 55 forgers that each of them tried to emulate 3 different genuine signatures 8 time each, in total 1,320 forged signatures.

The second one contains a collection of dutch signatures, some signers provided 24 original signatures, while others contributed with only 12 signatures, on the other hand the forgers created multiple forgery attempts , some with 8

signatures , others with 12 signatures, while some forgers did 16 attempts per signature.

Although this variation in the signatures makes the dataset more complicated, it also gives the model the adaptability and power to learn different kinds of signatures and effectively distinguish between authentic and fraudulent signatures across diverse writing styles.

Here are some examples of the signatures we worked on in this study :

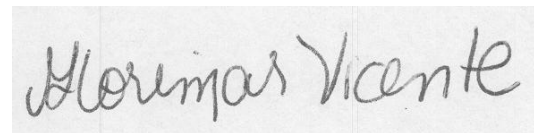


Fig 2 : genuine signature

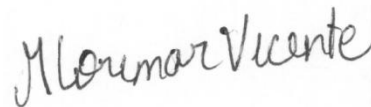


Fig 3 : forged signature

- **Image Preprocessing :**

Image processing involved a sequence of procedures to enhance the quality of the signature images, including techniques like noise reduction and thresholding and others, ensuring that our data is clean and ready for feature extraction and model training.

- **1-Gaussian Blur(Denoising):**

It is a well known image preprocessing technique designed to reduce high-frequency noise in images while protecting critical structures of the signature, using Gaussian kernel a mathematical function that applies a weighted smoothing approach to pixel neighborhoods.

Imagine you have a small 3x3 grid of pixels around a point in the image, the Gaussian function assigns each pixel in this grid a weight, with the central pixel receiving the highest weight, while the surrounding pixels receiving progressively lower weights according to a two-dimensional Gaussian distribution.

Then we calculate a weighted sum to replace the value of the center's pixel, this keeps the overall structure and decrease sharp changes caused by noise. Mathematically the Gaussian kernel is represented by the the function :

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where :

σ : Controls how much the blur spreads.

x, y : Coordinates of pixels in the grid relative to the center.

$2\pi\sigma^2$: The normalization factor to ensure the sum of all the weights equals 1, making it a proper average.

In this project a (3,3) kernel specifies a 3x3 grid of blurring, we calculated the weighted sum of the values based on the Gaussian function, then we replaced the central pixel with the result.

The choice of the small size of the kernel makes the blurring effect more localized, reducing the noise in the signature image, while maintaining important details in the signature.

2-OTSU Thresholding :

It is a probabilistic approach that calculates an optimal binarization threshold by minimizing intra-class variance, we use this mathematical technique to separate signature ink

from background, creating a binary representation that highlights signature's unique features.

Process of OTSU Thresholding :

1-Getting the frequency distribution of pixel intensities by computing the histogram of the image.

2-Calculating the total variance.

3-Iterate over all possible thresholds, by splitting the pixels into 2 groups, foreground that are pixels with intensity values less than or equal the threshold, and background that are pixels with intensity values greater than the threshold.

4-For each threshold t , the mean and the variance of the pixel intensities are calculated in foreground and background groups

5-Calculating the Within-Class variance for each threshold t with the function :

$$\sigma_w^2(t) = w_1(t) \cdot \sigma_1^2(t) + w_2(t) \cdot \sigma_2^2(t)$$

Where :

$w_1(t)$ and $w_2(t)$: the weights of the pixels in the foreground and background respectively.

σ_1^2 σ_2^2 : the variances of the pixel intensities in the foreground and background, respectively.

6-Finally the threshold that minimizes the within class variance is chosen, and the pixels with intensities greater than the threshold is assigned to foreground and the other is assigned to the background.

3-Laplacian sharpening :

This technique is applied to enhance edge details by employing 3x3 convolution kernel, with a central value

of 9 and -1 values around it, which makes the signature strokes more clear Laplacian kernel :

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

To sharpen an image using this kernel, for each pixel we multiply the kernel values with the surrounding pixel values and then we sum them up to get a new value for the center pixel.

This process results in a new image where edges become pronounced.



Fig 4 : Signature after preprocessing steps.

- **Feature extraction :**

Feature extraction plays a critical role in transforming raw signature images into meaningful information that can be used by machine learning models. By identifying and capturing distinctive characteristics of signatures, such as geometric, texture, and other numerical features, we can effectively represent each signature in a way that allows the model to learn and differentiate between genuine and forged signatures.

The feature extraction techniques used in our methodology are described in the section that follows.

1-Statistical features :

Ink density : We calculated in this the ratio of ink pixel in the image.

Mean intensity : The average pixel value in an image, it captures the overall brightness of the image.

$$\mu = \frac{1}{N} \sum_{i=1}^N I(x_i)$$

Where :

μ : The mean intensity

N : The total number of pixels

$I(x_i)$: The intensity of the i -th pixel in the image.

Standard intensity : measures how much the pixel intensities deviate from the mean, indicating the variation in brightness.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (I(x_i) - \mu)^2}$$

σ : The standard deviation of the pixel intensities

μ : The mean intensity

$I(x_i)$: The intensity of the i -th pixel in the image.

Skewness : the asymmetry of the pixel intensity distribution, skewness value of 0 indicates a perfectly symmetrical distribution, while positive or negative indicates skewness to the right or left.

$$\text{Skewness} = \frac{1}{N\sigma^3} \sum_{i=1}^N (I(x_i) - \mu)^3$$

μ : The mean intensity

σ : The standard deviation

$I(x_i)$ is the intensity of the i-th pixel

Kurtosis : describes the “fatness” of the tails found in probability distributions, there is three kurtosis categories, normal, less than normal and more than normal.

$$\text{Kurtosis} = \frac{1}{N\sigma^4} \sum_{i=1}^N (I(x_i) - \mu)^4 - 3$$

N: Total number of pixels in the image

μ : The mean intensity

σ : The standard deviation

$I(x_i)$ is the intensity of the i-th pixel

2-Geometric features :

Contour analysis allows us to extract essential shape elements from our signatures, and analyzing and evaluating those features allows us to better comprehend the signature's structure.

We extracted exactly three features :

The number of contours : The quantity of unique shapes or limits in the signature. In mathematical terms, it refers to the number of closed curves recognized in an image.

$$N_C = \text{len}(\text{contours})$$

Contour lengths : The entire distance around a form or border.

Mathematically, it is the contour's perimeter, which can be computed by adding the distances between adjacent points along the contour.

$$\text{Total Contour Length} = \sum_{i=1}^{N_C} P(C_i),$$

Where :

$P(C_i)$: The perimeter of contour C_i .

Contour areas : The space enclosed by the boundary. This can be computed by adding the areas of pixels within the contour. It provides an estimate of the size of the signature or certain sections of it.

$$\text{Total Contour Area} = \sum_{i=1}^{N_C} A(C_i),$$

Where :

$A(C_i)$: The area enclosed by contour C_i .

3-Texture features :

We applied the **Local Phase Quantization (LPQ)** technique to extract and analyze texture details in signatures. Each pixel in our signature holds an information. However, instead of analyzing individual pixels, we analyse the ‘Local window’, that is a 3x3 square of surrounding pixels that allows us to capture texture informations, with more details.

For a pixel (i,j) in and signature's image I , a local window of size 3x3 is :

$$W(i,j) = \{I(i-1,j-1), I(i-1,j), I(i-1,j+1), I(i,j-1), I(i,j), I(i,j+1), I(i+1,j-1), I(i+1,j), I(i+1,j+1)\}$$

Where :

$I(i,j)$: the intensity of the pixel at the position (i,j) .

For each local window, we calculated the **2D Fast Fourier Transform (FFT)**, we do this to change the data from the image space into the frequency space to

understand how the signature is made up of different frequency patterns like low or high frequency.

$$F(W(i, j)) = \mathcal{F}\{W(i, j)\}$$

Where :

$F(W(i, j))$ is the 2D frequency of the window.

\mathcal{F} : The fourier transform.

And when we extract the frequency data, we are interested on the phase, it is what tells us the patterns's shape, this phase holds the important details about the texture in the signature's image, and we calculate the phase for each 3x3 area to understand the texture better.

$$\theta(F(W(i, j))) = \arg(F(W(i, j)))$$

Where :

$\arg(F(W(i, j)))$: The phase angle of the 2D FFT.

Finally we quantize the phase values, we check if the phase of each frequency component is positive or negative, if it's positive we assign a value of 1, and if it's negative we assign a value of 0

Like that we make sure that the complexity of the phase information is reduced into a simple binary system.

$$Q(\theta(i, j)) = \begin{cases} 1, & \text{if } \theta(i, j) > 0 \\ 0, & \text{otherwise} \end{cases}$$

And after that we count the sum of the positive phase values, and this count give us the LPQ feature for the pixel.

$$LPQ(i, j) = \sum_{k=1}^9 Q(\theta(W(i, j))_k)$$

$\Theta(W(i, j))_k$: k-th phase angle in the 3x3

neighborhood.

And after doing all the work, we normalized the extracted features to prevent features with large values from dominating those with smaller values, we achieved this using either the Min-Max scaler or Standard Scaler, this ensures all features contribute equally during analysis.

Min-Max scaler : It scales the features into a fixed range, typically [0,1], this ensures all the features are on the same scale.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

It's ideal when preserving the relationships between feature values is important.

Standard scaler :

All features have a mean of 0 and a standard deviation of 1, which is important for algorithms that assume normally distributed data.

It's better for algorithms that assume normally distributed data or sensitive to variance.

$$x' = \frac{x - \mu}{\sigma}$$

Where :

μ : The mean

σ : The standard deviation

● Models :

SVM :

In our signature verification project ,we used Support Vector Machine (SVM),due to it's capacity to handle high-dimensional data and it's efficacy

in finding complex decision boundaries. After preprocessing the signature images and extracting relevant features, such as geometric features, texture features...

These features were then used to train the model to classify the signature either as genuine or forged.

This SVM algorithm aims to find an optimal hyperplane that maximizes the margin between two both classes.

SVM model was chosen due to it's versatility in kernels that provides a flexibility to adapt varying data characteristics

In our signature verification project, we applied a grid search to optimize the model's performance

In this project we applied a Min-Max and standard scaling to the data as a preprocessing step.

The best parameters we found when using Min-Max scaler were : {'C': 100, 'class_weight': None, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}

But when using Standard scaler the best parameters were : {'C': 100, 'class_weight': None, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}

Logistic regression :

Due to its simplicity, interpretability, and effectiveness in binary classification tasks, we used Logistic regression.

This algorithm was chosen for its ability to model the relationship between the features, also it is computationally efficient and performs well when the data is linearly separable.

The best parameters we found when using the Min-Max Scaler were: {'C': 100, 'max_iter': 500, 'solver': 'liblinear'}. Similarly, with Standard Scaling, the best parameters were: {'C': 1, 'max_iter': 500,

'solver': 'liblinear'}.

CNN :

Additionally, we explored the use of Convolutional Neural Networks (CNN) model combined with numerical features to improve the ability of signature classification.

The model used LPQ features we extracted earlier from signature images to capture texture details, also used the the different numerical features such as ink density, mean intensity, countor characteristics...

We fed the model preprocessed images resized to 256x256 with a single channel with normalized numerical features provided as a separate input.

The CNN architecture included three convolution layers with 16, 32 and 64 filters, each followed with by max-pooling layers for downsampling which reduced spatial dimensions and focus on important features, after applying the filter, we used an activation function ReLU, Rectified Linear Unit :

$$\text{ReLU}(x) = \max(0, x)$$

To prevent overfitting, dropout layers were applied with a dropout rate $p = 0.6$ where each neuron was randomly deactivated during training.

Also L2 regularization was used

$$\mathcal{L}_2 = \lambda \sum_i W_i^2$$

where λ is a regularization parameter and W_i is the weights of the model, ensuring that the model generalized well to unseen data.

After all the convolution and pooling layers, we flatten the output (turn it into a 1D vector) and pass it through fully connected layers.

A dense layer computes the output by multiplying the input by weights and adding a bias:

$$\text{Output} = W \cdot X + b$$

Where W is the weight matrix and b is the bias.

To convert the raw output values into probabilities, we used the 'softmax' function in the output layer

A dense layer with 'softmax' activation function served as the output layer, predicting the probabilities of the two classes : forged and genuine.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

For training part we employed the 'Adam' optimizer to update the model's weights, adjusting the learning rate using the past gradients and the squared gradients

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where

θ_t is the current weight

m_t is the average of gradients

v_t is the average of squared gradients

α is the learning rate

We added epsilon only to avoid division by 0

And to measure the difference between the predicted class probabilities and the actual labels we used categorical cross-entropy loss function

$$\mathcal{L}_{ce} = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

Where

C : the number of classes, in our case it's two

N : the number of samples

$y_{i,c}$: the actual label

$\hat{y}_{i,c}$: the predicted probability class c

And finally we applied an early stopping to prevent overfitting, in the case that the performance doesn't improve after some epochs, training stops before the last epoch.

• Performance :

After the training part, we evaluated the model on unseen data to assess its ability to generalize.

The result we got were kind of promising considering the complexity of the signature verification tasks.

We used the classification report to measure different metrics like precision, recall and F1-score.

Accuracy : how many accurate predictions there were out of all the predictions.

$$A = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100$$

Precision : It measures how many of the predicted positives are actually correct.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall : The proportion of true positive predictions to the total actual positives (all genuine signatures, both correctly and incorrectly predicted)

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-score : It provides a balance between the two, especially when there is a class imbalance. It's useful when you need to balance the trade-off between precision and recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Results :

Model	Train accuracy	Test accuracy
SVM (Standard scaling)	77.16%	72.96%
SVM (Min-Max scaling)	75.49%	71.92%
Logistic regression(Standard scaling)	66.14%	64.20%
Logistic regression(Min-max scaling)	66.33%	65.24%
CNN(Standard scaling)	86.25%	77.45%

• Conclusion :

In this study, we explored the verification signature project using both traditional machine learning techniques and deep learning models.

In this approach we successfully extracted some important features such

as geometric and texture features, alongside numerical features such as ink density and others that helped us know more details about each signature.

These features were after normalized using both Standard scaling and Min-Max scaling to ensure consistent input ranges for the models.

This preprocessing step was essential for improving model performance and ensuring a fixed range of [0,1], the models trained on these features included SVM for it's ability to classify high-dimensional data and CNN for their capability to learn complex image patterns.

While the result were promising, highlighting the potential of our current approach demonstrating the possibility of automated signature verification systems.

However, our approach faced limitations, including constraints related to computational resources, as well as the need for larger and diverse dataset to improve the model's capabilities, addressing these challenges in future work to improve the model's scalability and accuracy.

• Reference :

[1] The source code for this project is available on:

https://github.com/ESSAFI01/signature_verification

[2] A Comparative Study of Transfer Learning Models for Offline Signature Verification and Forgery Detection

[3] Offline Handwriting Signature Verification: A Transfer Learning and Feature Selection Approach

[4] Signature Verification Approach using Fusion of Hybrid Texture Features