Report: Online Signature Verification

2024JCS2040

Online signature verification is a biometric authentication method that analyzes the dynamic properties of a person's handwriting behaviour. Unlike static (offline) signatures, online signatures include temporal and pressure data such as pen coordinates, timestamps, and pressure levels.

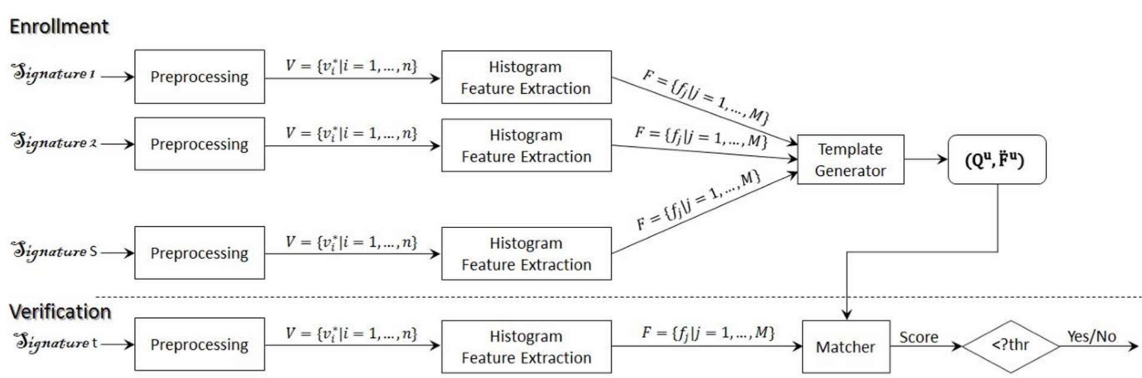
Dataset Overview : Used SVC2004 Dataset

Each signature sample is stored in a .TXT file named using the convention UxSy, where:

- U is a fixed character representing "User".
- x indicates the user ID (from 1 to N).
- S is a fixed character representing "Sample".
- y is the sample ID (from 1 to 40).

Samples 1–20 are genuine, while samples 21–40 are skilled forgeries. Each file contains:

- The first line: number of feature rows (e.g., 84)
- The next lines: space-separated values for:
  - X-coordinate
  - Y-coordinate
  - Timestamp
  - Button Status (0 for pen-up, 1 for pen-down)
  - Azimuth
  - Altitude
  - Pressure



Flow of the System

Steps:

Used libraries:

Numpy :Numerical ops, histograms, vector math

Pandas: CSV file creation, reading and writing tables

Os: File/folder management (Create output directories if they don't exist.)

Matplotlib: Visualizing score distributions and ROC

Sklearn: Evaluation: ROC, AUC, EER

1 Parsing and Preprocessing

Each .TXT file is read and the relevant 7-dimensional data is parsed. Files are converted into structured NumPy arrays and stored for further processing.

Output: .npy files per signature in Data_npy/step1_parsed_signatures/

And csv files are also generated to view the values in the .npy files

2 Vector Sequence Construction

For each parsed signature, these are computed:

- First and second order derivatives for X, Y, and Pressure.
- Polar transformations to compute magnitude (r) and angle (θ).

Each vector is represented as: vk = [xk, yk, rk, θk, pk]

Where :
x1 = 1st derivative of X (velocity in X-direction)

y1 = 1st derivative of Y (velocity in Y-direction)

r1 = 1st-order speed

theta(θ)1 = 1st-order movement direction

p1 = 1st derivative of pressure

x2 = 2nd derivative of X (acceleration in X-direction)

y2 = 2nd derivative of Y (acceleration in Y-direction)

r2 = 2nd-order speed

theta(θ)2 = 2nd-order acceleration direction

p2 = 2nd derivative of pressure (pressure acceleration)

These are concatenated into a final vector sequence.

Output: .npy vector files in Data_npy/step2_vector_sequences/

3 Feature Extraction

From each vector sequence, the following histograms are computed:

- 1D Histograms:
  - Angle (θ)
  - First derivative of angle (Δθ)
  - Magnitude (r)
- **2D Histograms (new addition):**
  - ⟨Φ1, R1⟩: relation between direction (angle) of stroke segment and speed (magnitude of movement) at segment
  - ⟨Φ1, ΔΦ1⟩: relation between direction and curvature

Each histogram is normalized and concatenated into a fixed-length feature vector. The inclusion of 2D histograms improves the separation between genuine and forged users by modeling inter-feature dependencies.

Starting I done without 2d histograms, then the matching score is less and EER is more compared to the inclusion of 2d histograms.

Output: .npy histogram feature vectors inData_npy/ step3_histogram_features/

4 Template Generation

For each user, a template is created using the genuine samples S1 to S20.

- Compute the mean vector of the features.
- Compute the standard deviation and derive the quantization step size.

(These all are calculated using the formulae given in the question)

Output: user_template.npy: Average feature vector. user_qstep.npy: Quantization step size.

Saved in: Data_npy/step4_templates/

5 Matching and Scoring

Match scores are computed for each test signature using:

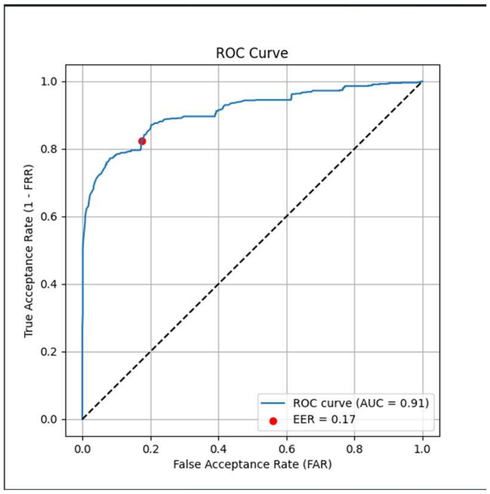
- Manhattan distance between quantized template and test vector
- Scores calculated for Genuine and Forgery signatures.
  - **Output**: match_scores.csv in Data_npy/step5_scores/

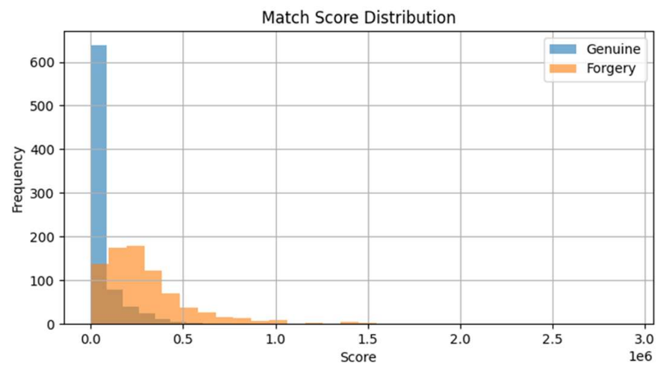
6 Evaluation and Analysis

- Histograms plotted to show score distribution for genuine and forgery

- ROC curve plotted
- Equal Error Rate (EER) computed
  EER (Equal Error Rate): point where False Acceptance Rate = False Rejection Rate.

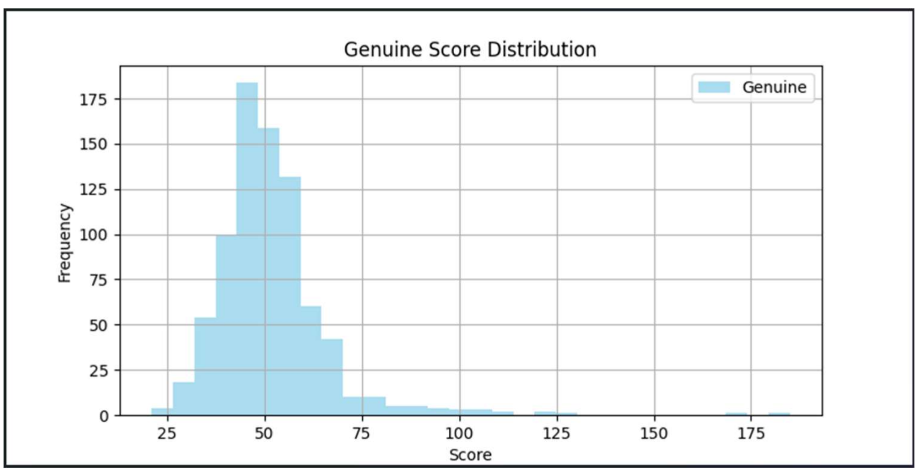
As I mentioned before Starting I only extracted 1D-histograms and generated the template, then these are results with 1D- Histograms.

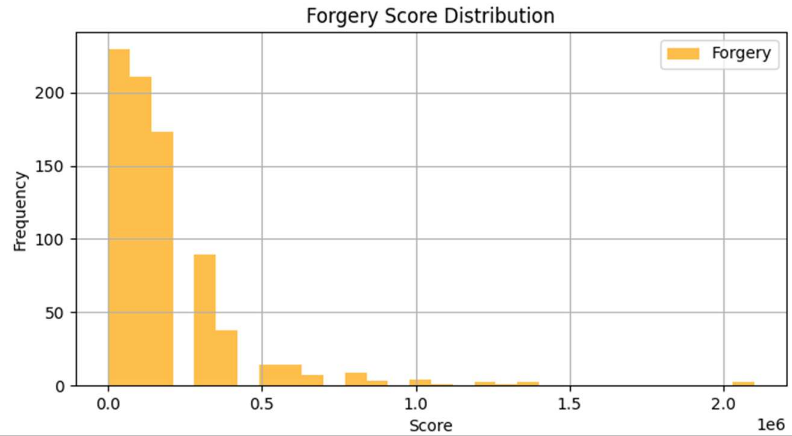




As we can see that the equal error rate is 0.17 and the area under roc curve is 0.91.
so this doesn't gave us less accuracy.

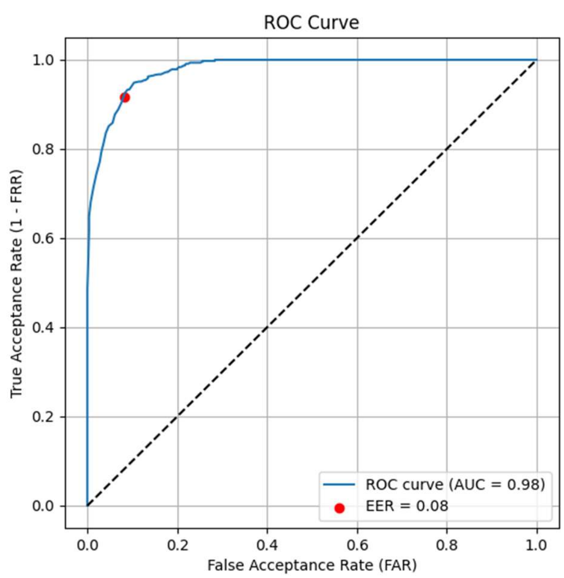
After Including 2D-histograms, the accuracy level as increased. These are the score distribution of genuine and forgery.





Genuine Min: 21.0, Max: 185.0, Mean: 51.5625

Forgery Min: 50.0, Max: 2100383.0, Mean: 179591.195



EER = 0.0825, Threshold = -67.00

This curve shows that the equal error rate is decreased and the AUC value is increased.
therefore the matching accuracy level is increased.

Signature Verification

```
jhancy@DESKTOP-UPF9MGV:/mnt/c/Users/VADDI JHANCY/desktop$ python3 verify.py
[INFO] Match score for U1S18: 42.00
[RESULT] ✅ Signature ACCEPTED as GENUINE
jhancy@DESKTOP-UPF9MGV:/mnt/c/Users/VADDI JHANCY/desktop$ python3 verify.py
[INFO] Match score for U1S38: 100084.00
[RESULT] ❌ Signature REJECTED as FORGERY
jhancy@DESKTOP-UPF9MGV:/mnt/c/Users/VADDI JHANCY/desktop$ python3 verify.py
[INFO] Match score for U2S38: 84.00
[RESULT] ❌ Signature REJECTED as FORGERY
jhancy@DESKTOP-UPF9MGV:/mnt/c/Users/VADDI JHANCY/desktop$ python3 verify.py
[INFO] Match score for U2S20: 41.00
[RESULT] ✅ Signature ACCEPTED as GENUINE
```

these are some outputs how it has correctly accepted the genuine ones and rejected the forgery ones.

This System provides a robust implementation of an online signature verification system using dynamic signature features. It effectively separates genuine and forgery attempts.

With the addition of **2D histogram features**, the system captures deeper relationships between motion dynamics (like direction-speed and direction-curvature), which improves classification boundaries and reduces overlap between genuine and forgery score distributions.

The final evaluation metrics, especially AUC and EER, demonstrate enhanced system performance.

Run the codes in this order:

1) python3 sig_parse.py -> outputs: step1_parsed_signatures
2) python3 deriva_polar.py -> outputs: step2_vector_sequences
3) python3 2d_histograms.py -> outputs: step3_histogram_features
4) python3 template.py -> outputs: step4_templates
5) python3 match_score.py -> outputs: step5_scores and EER and threshold values
6) python3 analysis.py -> outputs: creates roc curve and genuine, forgery scores
7) python3 verify.py -> outputs: verify if the input is genuine or not
8) python3 npy_view.py -> outputs: used to view the npy files of scores

sample inputs and outputs are in this link
https://drive.google.com/drive/folders/1ZklWMetPQrkPbGK9q3yWJvwKvu9ob77V?usp=sharing