Offline Signature Verification

BY

JIMMY DING JIA KANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Ts Dr Aun Yichiet, for his invaluable support, patience, time, and guidance in guiding me during my undertaking of this project. His practical suggestions and encouragement were crucial for the success of this project.

I would like to express my thanks to my parents, despite their financial constraints, funded the resources necessary to get my project off the ground.

I would also like to extend my thanks and appreciation to my lecturers in the FICT department who have taught me at one point or the other.

Next, to my friends who had given me practical criticism and advice on the project.

Finally, a heartfelt thank you to those who supported me, be it directly or indirectly, throughout the completion of this project.

Special thanks to Chong Jia Hui, Dennis Choong, Ho Jin Xin, Khor Jia En, Kuek Levin, Leow Yi Kang, Pang Jing Thean, Peggy Wong, and Yee Jia Hao for participating in my sample collection.

ABSTRACT

Handwritten signature remains a widely used behavioural biometric, supporting the importance of a robust offline signature verification. However, offline signature verification remains difficult due to high intra-class variability, high inter-class similarity, poor computational efficiency, and the imbalance of real-world training data. This project proposed a novel Convolutional Neural Network (CNN)-based model enhanced with L_{SC+} and an extension of the conventional PK sampling technique to address these limitations. L_{SC+} reduced the number of active triplets per batch by ignoring easy negatives, yielding comparable or better accuracy with fewer effective gradient updates. The model was trained and evaluated on the CEDAR benchmark dataset and further assessed using a self-sampled dataset. The proposed methodology aimed to improve computational efficiency and accuracy by placing greater emphasis on hard negatives and ignoring easier negatives. The loss function was designed to keep original signatures close together without over-optimisation. The proposed model was expected to achieve better performance and computational efficiency than the conventional Siamese networks or triplet loss function approaches. Due to resource constraints, all training was conducted using the recommended model, scheduler, and optimiser configurations. Without extensive hyperparameter fine-tuning, the proposed approach achieved up to 84.8% accuracy, 84.8% true positive rate, and an AUC score of 0.9284. Additionally, Grad-CAM visualisations were employed to qualitatively assess the model's decision-making process, revealing the presence of shortcut learning in a subset of cases while confirming that the model predominantly attends to stroke-based biometric features. These outcomes demonstrated the robustness and potential of the method, even under default settings. To the best of our knowledge, this was the first work to adapt L_{SC+}-style hard-negative-focused metric learning to offline signature verification and to introduce a $PKFM$ mini-batch sampler tailored to imbalanced genuine/forgery distributions.

Area of Study: Deep Learning, Offline Signature Verification

Keywords: Offline Signature Verification, Deep Metric Learning, Convolutional Neural Network, CEDAR, Intra-Class Signature, Inter-Class Signature, Loss Function

TABLE OF CONTENTS

TITLE PAGE **i**

COPYRIGHT STATEMENT **ii**

ACKNOWLEDGEMENTS **iii**

ABSTRACT **iv**

TABLE OF CONTENTS **v-viii**

LIST OF FIGURES **ix-x**

LIST OF TABLES **xi**

LIST OF SYMBOLS **xii**

LIST OF ABBREVIATIONS **xiii-xiv**

CHAPTER 1 INTRODUCTION **1-8**

1.1 Project Background 1-2

1.2 Problem Statements 2

1.3 Motivations 3

1.4 Project Scope 3-4

1.5 Project Objectives 5

1.6 Contributions 5-6

1.7 Report Organisation 7-8

CHAPTER 2 LITERATURE REVIEW **9-23**

2.1 Existing Implementations of Signature Verification Systems 9-16

2.1.1 Survey Paper on Deep Learning Signature Verification Systems 9-10

2.1.2 Offline Signature Recognition and Forgery Detection Using Deep Learning 10-11

2.1.3 Convolutional Neural Network (CNN) Based Offline Signature Verification Application 11

2.1.4 Enhancing Signature Verification Using Triplet Siamese Similarity Networks in Digital Documents 11-12

2.1.5 Learning Metric Features for Writer-Independent Signature Verification Using Dual Triplet Loss 12-14

2.1.6 A Two-Stage Siamese Network Model for Offline 14

Handwritten Signature Verification

2.1.7 Offline Signature Verification Using Region-Based Deep Learning Network 15-16

2.2 Loss functions 17-19

2.2.1 Hard Negative Examples are Hard, but Useful 17-18

2.2.2 FaceNet: A Unified Embedding for Face Recognition and Clustering 18-19

2.3 Convolutional Neural Network (CNN) Models 20-21

2.3.1 EfficientNetV2: Smaller Models and Faster Training 20-21

2.4 Limitation of Previous Studies 21-22

2.5 Proposed Solutions 22-23

CHAPTER 3 SYSTEM MODEL 24-33

3.1 Model Overview 24-25

3.2 Feature Extraction Backbone 25

3.3 Loss Function 26-28

3.4 Intuition 28-31

3.5 Methodologies and General Work Procedures 32-33

3.6 Timeline and Gantt Chart 34-35

CHAPTER 4 SYSTEM DESIGN 36-50

4.1 System Overview 36

4.2 Dataset Design and Preparation 36-42

4.2.1 CEDAR Dataset 36-38

4.2.2 Self-Sampled Dataset 38-39

4.2.3 Preprocessing of Signature Images 39-42

4.3 Sampling and Batch Construction 43-44

4.4 Model Architecture and Training Design 45-47

4.5 Inference and Analysis 48

4.6 Gradient-weighted Class Activation Mapping (Grad-CAM) 49-50

CHAPTER 5 EXPERIMENT/SIMULATION SETUP **51-67**

5.1 Hardware Setup 51

5.2 Software Setup 51-53

5.2.1 Development Environment and Primary Language 51

5.2.2 Deep Learning Framework and Libraries 52

5.2.3 Deployment Tools 53

5.3 Settings and Configurations 53-57

5.3.1 Distribution and Setup 53-54

5.3.2 Runtime Configurations 55-56

5.3.3 Deployment 56-57

5.3.4 License 57

5.4 System Operation 58-64

5.4.1 Preprocessing 58-59

5.4.2 Training and Evaluation 59-61

5.4.3 Inference and Demonstration 61-64

5.5 Implementation Issues and Challenges 65-66

5.6 Concluding Remark 67

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION **68-97**

6.1 System Testing and Validation Strategy 68

6.2 System Performance Metrics 69-70

6.3 Testing Setup 70-73

6.3.1 Evaluation Signature Images Setup 70-71

6.3.2 Threshold Selection 71-73

6.3.3 Grad-CAM 73

6.4 Training Observations (CEDAR Dataset) 74-83

6.4.1 Hard Triplet Strategy 74-77

6.4.2 Semi Hard Triplet Strategy 78-80

6.4.3 L_{SC} Loss Function 81-83

6.5 Performance and Objective Evaluation 84-86

6.5.1 Quantitative Results 84-85

6.5.2 Embedding Behaviour and Analysis 85-86

6.6 Grad-CAM 86-92
6.6.1 Shortcut Learning Effect 86-88
6.6.2 Classification Visualisations 89-92
6.7 Evaluation on Self-Sampled Dataset 92-95
6.7.1 Quantitative Results 92-93
6.7.2 Grad-CAM 94-95
6.8 Project Challenges 95-96
6.9 Concluding Remarks 96-97

CHAPTER 7 CONCLUSION AND RECOMMENDATION 98-100
7.1 Conclusion 98-99
7.2 Recommendation 99-100

REFERENCES 101-103
APPENDIX A i-ix
APPENDIX B x-xvi
POSTER xvii

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.2.2.1	Ideal Triplet	18
Figure 2.2.2.2	Hard Triplet	19
Figure 2.2.2.3	Semi Hard Triplet	19
Figure 2.3.1.1	Structure of MBConv and Fused-MBConv [11]	20
Figure 3.1.1	Model Overview	24
Figure 3.1.2	Grad-CAM Analysis Overview	25
Figure 3.2.1	Model Architecture	25
Figure 3.4.1	Embedding Space Dynamics Under L_{SC+} Loss Function	29
Figure 3.5.1	CRISP-DM Methodology	32
Figure 3.6.1	Project I Gantt Chart	34
Figure 3.6.2	Project II Gantt Chart	35
Figure 4.1.1	Overall System Design Pipeline	36
Figure 4.2.1.1	CEDAR Genuine Signature (9-1)	37
Figure 4.2.1.2	CEDAR Genuine Signature (25-11)	37
Figure 4.2.1.3	CEDAR Forged Signature (9-11)	37
Figure 4.2.1.4	CEDAR Forged Signature (25-5)	37
Figure 4.2.2.1	Self-Sampled Genuine Signature (2-1)	39
Figure 4.2.2.2	Self-Sampled Genuine Signature (5-1)	39
Figure 4.2.2.3	Self-Sampled Forged Signature (2-2)	39
Figure 4.2.2.4	Self-Sampled Forged Signature (5-1)	39
Figure 4.2.3.1	Example CEDAR Scan Before Preprocessing	40
Figure 4.2.3.2	Contrast Enhanced Sample	41
Figure 4.2.3.3	Gaussian Blurred Sample	41
Figure 4.2.3.4	Before Otsu's Thresholding	41
Figure 4.2.3.5	After Otsu's Thresholding	41
Figure 4.2.3.6	Folder Structured	42
Figure 4.2.3.7	Dataset Design and Preparation Subpipeline	42
Figure 4.4.1	Training and Validation Subpipeline	47
Figure 4.6.1	Grad-CAM Heatmap	50
Figure 5.3.1.1	Cloning GitHub Repository	53

Figure 5.3.1.2	Direct Download GitHub on Website	54
Figure 5.4.3.1	Interactive Deployment	61
Figure 5.4.3.2	Real-Time Model Switching	62
Figure 5.4.3.3	Threshold Slider	62
Figure 5.4.3.4	Genuine-Genuine Signature Pair	63
Figure 5.4.3.5	Genuine-Skilled Forgery Pair	63
Figure 5.4.3.6	Genuine-Easy Forgery Pair	64
Figure 5.4.3.7	False Positive Example	64
Figure 5.4.3.8	False Negative Example	64
Figure 6.3.1.1	Genuine Sample One	71
Figure 6.3.1.2	Genuine Sample Two	71
Figure 6.3.1.3	Genuine Sample One	71
Figure 6.3.1.4	Skilled Forgery Sample One	71
Figure 6.4.1.1	Hard Triplet Strategy Attraction Term Stacked Area Chart	74
Figure 6.4.1.2	Hard Triplet Strategy Repulsion Term Stacked Area Chart	74
Figure 6.4.1.3	Hard Triplet Strategy Attraction Term Histograms	75
Figure 6.4.1.4	Hard Triplet Strategy Repulsion Term Histograms	75
Figure 6.4.1.5	Hard Triplet Strategy ROC Curve	76
Figure 6.4.1.6	Hard Triplet Strategy Confusion Matrix	77
Figure 6.4.2.1	Semi Hard Triplet Strategy Attraction Term Stacked Area Chart	78
Figure 6.4.2.2	Semi Hard Triplet Strategy Repulsion Term Stacked Area Chart	78
Figure 6.4.2.3	Semi Hard Triplet Strategy Attraction Term Histograms	79
Figure 6.4.2.4	Semi Hard Triplet Strategy Repulsion Term Histograms	79
Figure 6.4.2.5	Semi Hard Triplet Strategy ROC Curve	80
Figure 6.4.2.6	Semi Hard Triplet Strategy Confusion Matrix	80
Figure 6.4.3.1	L_{SC+} Attraction Term Stacked Area Chart	81
Figure 6.4.3.2	L_{SC+} Repulsion Term Stacked Area Chart	81
Figure 6.4.3.3	L_{SC+} Attraction Term Histograms	81
Figure 6.4.3.4	L_{SC+} Repulsion Term Histograms	81

Figure 6.4.3.5	L_{SC+} ROC Curve	83
Figure 6.4.3.6	L_{SC+} Confusion Matrix	83
Figure 6.6.1.1	Signer 19 Shortcut Learning	87
Figure 6.6.1.2	Signer 21 Shortcut Learning	87
Figure 6.6.1.3	Signer 19 Genuine Signature	88
Figure 6.6.1.4	Signer 52 Genuine Signature	88
Figure 6.6.1.5	Signer 52 Genuine Signature	88
Figure 6.6.1.6	Signer 52 Forged Signature	88
Figure 6.6.2.1	True Positive Grad-CAM Example One	89
Figure 6.6.2.2	True Positive Grad-CAM Example Two	90
Figure 6.6.2.3	True Negative Grad-CAM Example One	90
Figure 6.6.2.4	True Negative Grad-CAM Example Two	90
Figure 6.6.2.5	False Negative Grad-CAM Example One	91
Figure 6.6.2.6	False Negative Grad-CAM Example Two	91
Figure 6.6.2.7	False Positive Grad-CAM Example One	92
Figure 6.6.2.8	False Positive Grad-CAM Example Two	92
Figure 6.7.1.1	Confusion Matrix	93
Figure 6.7.2.1	Signer 5 Grad-CAM Pair 1	95
Figure 6.7.2.2	Signer 5 Grad-CAM Pair 2	95
Figure 6.7.2.3	Signer 5 Grad-CAM Pair 3	95
Figure 6.7.2.4	Signer 5 Grad-CAM Pair 4	95
Figure 6.7.2.5	Signer 5 Grad-CAM Pair 5	95
Figure 6.7.2.6	Signer 5 Grad-CAM Pair 6	95
Figure A1	Signer 1 Genuine 1	I
Figure A2	Signer 1 Genuine 2	i
Figure A3	Signer 1 Genuine 3	i
Figure A4	Signer 1 Genuine 4	i
Figure A5	Signer 1 Forged 1	ii
Figure A6	Signer 1 Forged 2	ii
Figure A7	Signer 1 Forged 3	ii
Figure A8	Signer 1 Forged 4	ii
Figure A9	Signer 2 Genuine 1	iii
Figure A10	Signer 2 Genuine 2	iii

Figure A11	Signer 2 Genuine 3	iii
Figure A12	Signer 2 Genuine 4	iii
Figure A13	Signer 2 Forged 1	iii
Figure A14	Signer 2 Forged 2	iii
Figure A15	Signer 2 Forged 3	iii
Figure A16	Signer 2 Forged 4	iii
Figure A17	Signer 3 Genuine 1	iv
Figure A18	Signer 3 Genuine 2	iv
Figure A19	Signer 3 Genuine 3	iv
Figure A20	Signer 3 Genuine 4	iv
Figure A21	Signer 3 Forged 1	v
Figure A22	Signer 3 Forged 2	v
Figure A23	Signer 3 Forged 3	v
Figure A24	Signer 3 Forged 4	v
Figure A25	Signer 4 Genuine 1	vi
Figure A26	Signer 4 Genuine 2	vi
Figure A27	Signer 4 Genuine 3	vi
Figure A28	Signer 4 Genuine 4	vi
Figure A29	Signer 4 Forged 1	vii
Figure A30	Signer 4 Forged 2	vii
Figure A31	Signer 4 Forged 3	vii
Figure A32	Signer 4 Forged 4	vii
Figure A33	Signer 5 Genuine 1	viii
Figure A34	Signer 5 Genuine 2	viii
Figure A35	Signer 5 Genuine 3	viii
Figure A36	Signer 5 Genuine 4	viii
Figure A37	Signer 5 Forged 1	ix
Figure A38	Signer 5 Forged 2	ix
Figure A39	Signer 5 Forged 3	ix
Figure A40	Signer 5 Forged 4	ix
Figure B1	Pair 1	x
Figure B2	Pair 2	x
Figure B3	Pair 3	xi

Figure B4	Pair 4	xi
Figure B5	Pair 5	xii
Figure B6	Pair 6	xii
Figure B7	Pair 7	xiii
Figure B8	Pair 8	xiii
Figure B9	Pair 9	xiv
Figure B10	Pair 10	xiv
Figure B11	Pair 11	xv
Figure B12	Pair 12	xv
Figure B13	Pair 13	xvi

LIST OF TABLES

Table Number	Title	Page
Table 2.1.1.1	Offline Signature Datasets Comparison [1]	9
Table 3.5.1	Activities to be Completed	33
Table 4.2.1.1	Dataset Splitting	38
Table 4.4.1	Comparison Between EfficientNetV2 Variants [11]	45
Table 4.4.2	Checkpoint Saved State	46
Table 4.5.1	Summary of Methods	48
Table 4.6.1	Technical Configuration	50
Table 5.1.1	Specification of Computer	51
Table 5.2.1.1	Development Environment	51
Table 5.2.2.1	Deep Learning Framework and Libraries	52
Table 5.2.3.1	Deployment Tools	53
Table 5.3.2.1	Configurable Training Parameters	55
Table 5.3.2.2	Optimiser Parameters	56
Table 5.3.2.3	Schedulers Parameters	56
Table 5.3.2.4	Hyperparameter Choice Rationale	57
Table 6.3.2.1	Threshold Selection	72
Table 6.4.1.1	Hard Triplet Strategy Evaluation Results	75
Table 6.4.2.1	Semi Hard Triplet Strategy Evaluation Results	79
Table 6.4.3.1	L_{SC+} Evaluation Results	82
Table 6.5.1.1	Comparison of Results	84
Table 6.7.1.1	Evaluation Results (Self-Sampled)	93

LIST OF SYMBOLS

$\sum$	Summation over training samples or time steps	Unitless
λ	Weighting factor between skilled and random losses	Unitless
Σ_τ	Summation over training samples or time steps	-(operator)
τ	Index over training samples or time steps	-(index)
S_{ap}	Similarity score between the anchor and a positive sample	Unitless
S_{an}	Similarity score between the anchor and a negative sample	Unitless
L	Loss value	Unitless
L_{SC+}	Selectively Contrastive Loss Plus	Unitless

LIST OF ABBREVIATIONS

Adam	Adaptive Moment Estimation
AdamW	Adam with Decoupled Weight Decay
AUC	Area Under the Curve
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
EER	Equal Error Rate
FAR	False Acceptance Rate
FP	False Positive
FRR	False Rejection Rate
Grad-CAM	Gradient-Weighted Class Activation Mapping
HN	Hard Negative
HoG	Histogram of Orientated Gradients
LBP	Local Binary Patterns
LR	Logistic Regression
L_{SC+}	Selectively Contrastive Plus Loss Function
MSDN	Mutual Signature DenseNet
PDPA	Personal Data Protection Act
ROI	Region of Interest
ROC	Receiver Operating Characteristic
SCT	Selectively Contrastive Triplet Loss
SE-Blocks	Squeeze-and-Excitation blocks

SGD	Stochastic Gradient Descent
SHN	Semi-Hard Negative
SPP	Spatial Pyramid Pooling
SSN	Siamese Similarity Network
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
tSSN	Triplet Siamese Similarity Network
TP	True Positive
WD	Writer-Dependent
WI	Writer-Independent

CHAPTER 1 INTRODUCTION

In this chapter, the project background, problem statements, motivations for the project, objectives, project scope and direction, and contributions to the field of offline signature verification, and the outline of the thesis are presented.

1.1 Project Background

Handwritten signature verification remains a critical research area, particularly given its persisting importance in legally binding documents. Although digital signatures have gained traction, they are not yet universally accepted; certain legal documents, such as wills, trusts, negotiable instruments, and powers of attorney, frequently mandate a physical handwritten signature to be considered valid. Offline handwritten signatures, as a result, continue to play a crucial role in legal and financial contexts, and underscore the need for robust, automated verification systems. Nevertheless, the process remains largely manual. This manual approach is not only time-intensive but also prone to human error and fatigue.

This led to the development of automated signature verification systems; these streamline the process by allowing signatures that fall within a certain threshold to pass [3], drastically reducing the volume of documents requiring manual review. However, these systems still struggle with the natural variations and inconsistencies inherent in human handwriting [2].

Signature verification systems can be divided into two categories: online and offline. In online signature verification systems, signatures are obtained via an electronic input device, such as a graphics tablet or stylus-equipped touchscreen. These devices capture not only the final image but also dynamic temporal information–acceleration (how quickly the pen moves), speed (the rate of signature creation), pressure (the force applied during writing), and pen angle (the tilt of the writing instrument). In contrast, offline signature verification systems rely on static, scanned signatures obtained from documents. Due to this collection method, the dynamic characteristics of signatures are lost, making verification more challenging.

Early approaches to automatic signature verification primarily relied on geometric and statistical methods such as Histogram of Oriented Gradients (HoG) and Local Binary Patterns (LBP). However, recent research trends are leaning toward the use of deep

learning techniques for enhanced accuracy and adaptability [3]. This shift in techniques introduced new problems to address, including the inherent low-shot learning nature of the problem, high computational costs, and high error rates.

1.2 Problem Statements

Errors Due to Human Fatigue When Verifying Large Amounts of Documents

Current verification workflows still rely heavily on human experts to distinguish genuine signatures from forgeries. However, manual review is highly susceptible to cognitive fatigue and exhaustion, especially when processing high volumes of documents. In high-stakes financial, legal, and administrative contexts, the resulting increase in error rates can lead to dire consequences. By introducing a reliable offline signature verification system, document volumes can be filtered, reducing the volume requiring manual verification and saving time and energy.

Struggle with Intra-Class Variability and Skilled Forgeries

An effective model must be able to generalise to the inherent variations in signatures while simultaneously detecting skilled forgeries. Current implementations of offline signature verification systems struggle with these objectives. This struggle can be attributed to the intrinsic nature of offline signature verification problems; they are few-shot learning problems. Deep learning models are prone to overfitting on the limited genuine samples available. This demands a more specialised architecture and a refined loss function to address the limited data and prevent overfitting.

Computational Efficiency

The conventional triplet loss function implemented in offline signature verification systems suffers from poor optimisation, inefficiencies that waste computational resources. Specifically, standard triplet loss continues to apply gradients to triplets that already have the optimal positioning (the positive samples are already closer to the anchors than the negative samples). This redundant computation leads to over-optimisation of the anchor-positive distance, which collapses the embedding space and may render the model unable to generalise to natural variations or to unseen signer styles.

1.3 Motivations

The motivation for this project stems from the persisting importance of offline signatures in legally binding documents, such as wills, trusts, and codicils. Despite the emergence of biometric and digital authentication, physical signatures hold legal and cultural significance. Current manual verification processes are not only labour-intensive but are also highly susceptible to human error. These errors create bottlenecks: False Positives allow for fraudulent transactions, while False Negatives obstruct legitimate legal proceedings. Automating this process and reducing document volume are therefore essential to mitigate the dire financial and legal consequences of erroneous human judgement.

While the integration of deep learning models into offline signature verification pipelines is promising, existing systems often struggle with poor performance and computational inefficiency. This is primarily due to the inherent low volume of offline signature images [1] and the reliance on generalised loss functions that are not optimised for the nuances of signatures, such as the conventional triplet loss function. The core motivation of this project lies in engineering a specialised loss function designed to overcome these constraints, creating a model that is robust enough to pave the way for future research.

Finally, this project is driven by the imperative for computational efficiency. By exploring a decoupled loss formulation, this project aims to accelerate model training, reducing hardware requirements and enabling an offline verification system accessible to a wider range of constraints and applications.

1.4 Project Scope

In Scope

The primary aim of this project was to develop a deep learning model utilising transfer learning from the EfficientNet-V2 architecture. A central component of the scope was the design and implementation of a specialised loss function, referred to as the Selectively Contrastive Loss Plus, L_{SC+}. This loss function was adopted, adapted, and extended from [2]. To validate the efficacy of this proposed loss function, its the model trained on this loss function was benchmarked against models trained with hard and semi-hard triplet mining. Additionally, to aid with the lack of training data, an

extension of PK sampling, $PKFM$ was created, providing us with more control over the distribution of training samples.

Out of Scope

To prevent scope creep, the following areas were excluded from this project:

- **Online Handwritten Signatures**
  - This project was strictly limited to offline handwritten signatures. It did not cover online or digitally generated signatures created with e-signature tools, as these involve temporal biometric data warrant fundamentally different verification techniques.
- **Script Limitations**
  - This project was confined to Latin-based scripts such as English, Spanish, or Malay. Logographic systems (e.g. Chinese, Japanese, and Korean) and Brahmic scripts (e.g. Tamil and Bengali) were excluded due to their distinct morphological and structural characteristics.
- **Image Quality Influence**
  - This project did not address accuracy issues caused by poor-quality images, heavy occlusion, or corrupted scans.
- **Signature Extraction**
  - This project did not include extracting signatures from documents. It was assumed that the signatures have already been localised and isolated from the documents.
- **Writer-Dependent Verification**
  - This project was only focused on writer-independent verification. This meant that one model was trained across multiple signers, rather than writer-dependent verification, where one model is trained for each signer.

1.5 Project Objectives

The primary objective of this project was to develop and evaluate an offline signature verification model that balances high discriminative accuracy with computational efficiency. Success was measured using evaluation metrics such as AUC Score, Equal Error Rate (EER), and overall accuracy to ensure the model met the performance requirements.

The objectives are detailed as follows:

1. **To develop a writer-independent signature verification model**
    a. Incorporated EfficientNetV2 as the backbone to generate high-dimensional embeddings from offline signature images.
    b. Ensured model generalisation by learning signature features rather than memorising specific identities.
2. **To design and implement a specialised loss function, L_{SC+}, for offline handwritten signatures**
    a. Adapted Selectively Contrastive Triplet Loss, L_{SC}
    b. Enforced explicit positive-distance constraints to ensure genuine signatures from the same writer are clustered in the embedding space.
3. **Designed a $PKFM$ mini-batch sampling strategy to balance intra- and inter-class comparisons.**
    a. Extended PK sampling to control writer, genuine, and forgery distributions within each batch.
    b. This explicit control provided a balanced ratio of intra-class and inter-class comparisons.

1.6 Contributions

This project addressed core challenges in the offline signature verification domain: the high intra-class variability in no two signatures from the same signer and the lack of dynamic features like stroke order, timing, or pressure, which limited verification to only the aftermath of the strokes. By creating a model capable of handling natural variations without compromising its discriminative power against skilled forgeries, this project facilitated by introducing a reduction in manual verification workloads.

The specifics of the contributions of this work are as follows:

- Development of a specialised loss function, L_{SC+}, for offline signature verification
  - This project used the L_{SC} loss function from [2], which had yet to be adapted to the domain of offline signature verification. The loss function was extended into L_{SC+}, where a positive-pull term was introduced to stabilise genuine clusters, ensuring generalisability even under high intra-class variability.
- $PKFM$ mini-batch sampler
  - We proposed a $PKFM$ sampler that constructed each mini batch with a controlled number of signers:
    - P signers and K genuine samples per signer.
    - F skilled forgeries provide hard negatives.
    - M additional inter-signer negatives to provide global contrast
  - These changes directly addressed class imbalance and ensured that every gradient update was formed by a balanced mix of intra-class and inter-class comparisons.
- Comprehensive writer-independent evaluation on CEDAR
  - We conducted a writer-independent evaluation on the CEDAR dataset, benchmarking the proposed L_{SC+} against standard hard and semi-hard triplet losses variants, all of which used the same EfficientNetV2 backbone. The proposed model achieved 84.8% accuracy and an AUC of 0.9284.

In low-shot learning scenarios such as signature verification, where data volume is often suboptimal, maximising the value of every sample is important. This project demonstrates how a specialised loss function can learn an efficient embedding space that generalised loss functions fail to learn.

Finally, this work provide a foundational framework for future research on loss functions for tasks that share characteristics with the offline signature verification domain.

1.7 Report Organisation

This report is structured into seven chapters that detail the conceptualisation, implementation, and evaluation of the proposed offline signature verification framework.

Chapter 1: Introduction

The project's background, problem statements, motivations, scopes, and specific objectives are introduced. It also discusses the project's contributions to the wider field and provides an overview of the entire report structure.

Chapter 2: Literature Review

This chapter presents a comprehensive literature review covering existing research on various implementations, CNN models, and loss functions. Here, the strengths and limitations highlighted by previous research are explored and serve as a foundation for identifying potential avenues for exploration and innovation.

Chapter 3: System Model

The description of the architecture behind the custom loss function with equations and examples. This chapter also provides the reader with scenarios with a description of how the model would respond.

Chapter 4: System Design

The pipeline design is presented in this chapter, with technical details independent of any tools or services. This chapter highlights the logic behind implementation decisions and design choices without limiting them to specific tools.

Chapter 5: Experiment/Simulation

Provides details on the implementation of the training, evaluating and testing pipeline, providing technical understanding of the developed framework. These details are to ensure that the implementation and testing of this project can be reproduced by interested third parties. This includes the hardware and software environment during development.

Chapter 6: Results and Discussion

Focuses on system evaluation and discussion, where metrics such as accuracy, True Positive Rate, False Positive Rate, Area Under the Curve Score and Equal Error Rate (EER) are measured. These results are analysed and compared to evaluate the model's effectiveness in meeting the defined objectives.

Chapter 7: Conclusion

Concludes the project with a summary of achievements and reflections on potential limitations. It also discusses the model's real-world applicability and suggests directions for future enhancements in the domain.

CHAPTER 2 LITERATURE REVIEW

2.1 Existing Implementations of Signature Verification Systems

2.1.1 Survey Paper on Deep Learning Signature Verification Systems

In general, signature verification involves four main stages: data acquisition or digitisation of handwritten signatures, preprocessing of the acquired signatures, feature extraction for the statistical approach or structural representation used in structural approach, and classification of the provided signature [1, 3]. Due to the difference in nature, offline and online signatures have different verification methods and datasets. The most used offline signature datasets are shown in Table 2.1.1 [1].

Table 2.1.1.1 Offline Signature Datasets Comparison [1]

Dataset name	Language	Number of Signers	Genuine	Forged	Total Signatures
CEDAR	Multiple backgrounds	55	24	24	2640
MCYT-75	Spanish	75	15	15	2250
GPDS-syntheses	Computer-generated		24	30	4000
SigComp`11	Dutch	64			1932
SigComp`11	Chinese	20			1177
SigWiComp`13	Japanese	30	42	36	2340

The existing implementations are further distinguished by the method in which the signatures are represented in the system (statistical or structural). Structural representations (graph-based and contour-based models), generally achieve better verification accuracy compared to statistical. However, these approaches typically require greater computational resources and larger storage and result in slower response times compared to feature-embedding methods [3]. The core of statistical representation is the local or global extraction of features from the signature. The local scale focuses on analysing sub-regions of the signature images, capturing independent details such as stroke patterns, textures, or localised distortions, whereas the global scale analyses the entire signature image to produce a single feature vector [3].

These implementations can be categorised as either writer-dependent or writer-independent, with the latter being more prevalent. In a writer-dependent implementation, the system implements a separate model for each signer, resulting in higher storage requirements and computational power demands. In contrast, writer-independent approaches train a single model on the entire dataset, offering greater flexibility [1]. Since signature verification is ultimately a classification problem, its performance can be evaluated using the false rejection rate (FFR, Type I error) and false acceptance rate (FAR, Type II error), both of which are shown in equation 2.1.1.1 and equation 2.1.1.2. These metrics can be further used to create a receiver operating characteristic (ROC) curve and determine the equal error rate (EER) [3].

$$\text{FRR} = 1 - \frac{\text{TP}}{\text{number of genuine signatures}} \#$$

$$\text{FAR} = \frac{\text{FP}}{\text{number of forged signatures}} \#$$

Deep metric learning is a common approach to train deep learning models in signature verification. As a result, the selection of distance measures has significant impact on the accuracy of how the embedding space is shaped. Due to their ease of use and simplicity for small or isolated clusters, Euclidean and Manhattan distances are two of the commonly most used metrics. However, both metrics are sensitive to outliers [2].

2.1.2 Offline Signature Recognition and Forgery Detection using Deep Learning

[4] demonstrated the efficacy of a system that combines both deep learning and classical feature-based approaches. The authors suggested a hybrid strategy that uses a Crest-Trough algorithm for forgery detection and a Convolutional Neural Network (CNN) for signature verification, with post-checks using the Harris corner detection algorithm and Speeded-Up Robust Features (SURF). Prior to feature extraction and CNN processing, they isolate and align the signature region of interest (ROI) using preprocessing techniques such as noise removal, scaling, centralisation, and rotation normalisation. Strong empirical performance is reported, with 95% accuracy in signature recognition and 85% to 89% forgery detection rates.

Nonetheless, this study does not explore metric-learning techniques, Siamese networks, triplet embedding strategies, or robustness experiments against image

distortions; instead, it is restricted to a CNN in conjunction with a feature-engineering pipeline. Therefore, it serves as a useful baseline for future research and this project.

2.1.3 Convolutional Neural Network (CNN) Based Offline Signature Verification Application

CNNs are well-suited for signature verification tasks because they can automatically extract discriminative local and global spatial features, reducing reliance on handcrafted feature engineering [5]. The authors evaluated both Writer-Dependent (WD), in which models are trained independently for each signer, and Writer-Independent (WI), in which a single model is trained across numerous signers, in their ad hoc CNN-based technique for offline signature verification. The GPDS Synthetic Signature dataset, which includes 4,000 signers with both real and fake signatures, was used by the authors to train their models. Greyscale photos reduced to 210x300 were fed into the CNN architecture, which included five convolutional layers, two pooling layers, three dense layers, and dropout. The accuracy of the WD models was 75%, whilst the accuracy of the WI models was 62.5%, according to the results. Although their ad hoc CNN worked rather well, the authors pointed out that performance was constrained by the lack of supplementary feature extraction techniques, indicating that hybrid approaches might produce better results.

2.1.4 Enhancing Signature Verification Using Triplet Siamese Similarity Networks in Digital Documents

[6] proposed their implementation of a triplet Siamese similarity network (tSSN) to authenticate signatures. Euclidean, Manhattan, and Minkowski distances were among the distance metrics they experimented with. Before inputting signatures into a lightweight CNN backbone, they preprocess them using bounding-box ROI extraction, greyscale conversion, and normalisation. The model learns an embedding space where counterfeit signatures are pushed apart and authentic signatures cluster closely by training on triplets (anchor, positive, and negative). The last classification is then carried out using a two-layer similarity network. Their tSSN design performed better than both a standalone CNN and a conventional Siamese network.

Using FAR and FRR as assessment criteria, the authors assessed their model on a few benchmark datasets, including BHSig260, SigComp2011, 4NSigComp2010, 4NSigComp2012, CEDAR, MCYT, and GDPS. The model consistently achieved

higher accuracy ratings across these datasets, outperforming both ordinary Siamese networks and conventional CNNs. For instance, tSSN reported an F1-score of 90.1% on 4NSigComp2010, while CNN and Siamese-based CNN recorded scores of 60.5% and 75.2% respectively. Robustness testing, however, revealed a significant drawback: the system's performance dropped to the 55–65% range when was subjected to geometric modifications such as rotation, scaling, and flipping.

Furthermore, the triplet loss function outperformed contrastive loss and binary cross-entropy loss. For instance, the triplet loss function had an accuracy of 92.2% at SigComp2011, compared to 76.1% for contrastive loss and 70.2% for binary cross-entropy. Additionally, the authors' implementation also surpassed other pre-trained models consistently across multiple datasets, including VGG16, ResNet, MobileNet, and EfficientNet. Moreover, tSSN also consistently beat machine learning algorithms such as Support Vector Machine (SVM), Logistic Regression (LR), and Gradient Boosting (XGBoost) Classifier.

Ultimately, it was discovered that the Manhattan distance metric outperformed the Euclidean distance and Minkowski distance metrics. Nevertheless, the authors noticed that eliminating any one of the model's components would result in a notable decline in performance across all datasets.

2.1.5 Learning Metric Features for Writer-Independent Signature Verification using Dual Triplet Loss

In 2021, Q. Wan and Q. Zou introduced the use of dual triplet loss for a writer-independent signature verification solution [7]. The fundamental idea behind triplet loss is still to widen the distance between references and forged signatures, while narrowing the gap between references and authentic signatures. Their novelty lies in the system's capacity to distinguish between experts and random forgeries; skilled forgeries should be more similar to the reference signature than random forgeries. They calculated the loss function using a Convolutional Neural Network (CNN) with a Euclidean distance metric, as shown in equation (2.1.5.1).

$$L=\Sigma_{\tau}[\lambda \times L_{\text{skilled}}+(1\text{-}\lambda) \times L_{\text{random}}]$$

A weighted loss function that balances contributions from experts and random forgeries is defined by equation (2.1.5.1). The sum of all training samples, τ,

determines the overall loss, L, where λ regulates the relative importance of L_{skilled} and L_{random}.

They have employed a batch hard method and soft margin, eliminating any invalid or readily satisfied triplets during training, in order to alleviate the computational cost. When the desired distance constraint is met, the loss would be zero if the author had not employed a soft margin. This is problematic as it results in zero gradients, which prevents the model from learning any further. The model will still be penalised with the implementation of a soft margin, albeit at a reduced rate, even when the primary margin is satisfied. As a result, the model is made to continue learning, decreasing the difference between authentic and reference signatures and maximising the difference between fake and reference signatures.

To assess the model's efficacy and generalisability, the authors pre-trained it on a large synthetic dataset, the GPDS dataset, and then used transfer learning to refine it on real-world datasets, the MCYT and CEDAR datasets. By evaluating their model against a state-of-the-art baseline under identical experimental conditions, ensuring fair comparison. The Area Under the Curve (AUC), False Rejection Rate (FRR), False Acceptance Rate (FAR), and the Equal Error Rate (EER) are a few common metrics that they authors used to evaluate their model. With a best EER of 7.34% with 12 reference samples, their model outperformed the previous state-of-the-art SigNet-F model on the GPDS Synthetic dataset. On the CEDAR dataset, the pre-trained model was very competitive. After fine-tuning, it performed well, attaining near-perfect 0.0% EER. The model still performed well on the MCYT, reporting a best EER of 12.83%. The authors hypothesised that the dataset distributions may have caused the performance discrepancy. Compared to conventional writer-dependent classifiers like Support Vector Machines (SVMs), their approach significantly reduces processing cost.

The SigCNN model outperformed the SigNet-F model in terms of generalisation abilities when tested on the Chinese dataset CSIG-WHU. However, both models performed worse on the Chinese dataset than on the CEDAR and MCYT datasets, and fine-tuning with a small number of users made very little differences. They conjectured that the decline in performance might be caused by the differences in characteristics between character-based and alphabet-based signatures. Furthermore,

the training benchmark datasets are often highly controlled and do not accurately reflect the natural variations present in real-world signatures, such as variations in pen thickness or slight shifts in a person's writing style.

2.1.6 A Two-Stage Siamese Network Model for Offline Handwritten Signature Verification

Xiao and Ding [8] addressed the challenges of class imbalance and weak feature representation in offline handwritten signature verification by proposing a two-stage Siamese neural network (SNN). Their architecture featured an extra image-enhancement branch intended to boost discriminative writer-style features prior to comparison. The authors adopted focal loss instead of standard cross-entropy to overcome the significant class imbalance between positive and negative pairs. This method reduces any potential dominance of the majority class while increasing the model's sensitivity to difficult-to-classify samples.

Two parallel Siamese paths made up the model. The first processed the original signature pair directly, while the second processed an improved version created by a lightweight attention-like module. In particular, this enhancement path created a weight matrix by using global average pooling on intermediate CNN feature maps, followed by fully connected layers, reshaping, and up sampling. To highlight salient regions, this matrix was multiplied by the original signature image. The model was able to simultaneously optimise raw and enhanced representations for better distriminative capability since both paths were trained using focal loss and their individual losses were merged via a weighting factor.

CEDAR, BHSig-Bengali, BHSig-Hindi, and a Chinese forensic signature dataset were used to assess the method in a writer-independent scenario. The results denoted better accuracy than baselines like SigNet, morphological characteristics, and ensemble learning techniques, with over 90% on BHSig-Bengali and 95.7% on CEDAR. Both single-stage SNN and image-enhanced SNN performed worse than the suggested two-stage design, with ablation tests validating the benefit of combining the enhancement branch with focal loss. The introduction of a two-stage Siamese network with feature enhancement, the use of focal loss to overcome class imbalance in the HSV dataset, and an evaluation on a novel real-world Chinese signature dataset are the main contributions of this study.

2.1.7 Offline Signature Verification Using Region-Based Deep Learning Network

Li Liu, Linlin Huang, Fei Yin, and Youbin Chen [9] proposed an approach that segments each signature into multiple overlapping regions, extracts local feature representations, and computes similarity scores using a region-based metric-learning model. The similarity scores from each region were fused to create a final classification.

The authors applied pre-processing techniques such as modifying the signature's angle, eliminating the backdrop, and normalising the foreground greyscale values and image dimensions. Additionally, if the signature images were coloured, the authors made sure they were converted into greyscale using Otsu's binarisation method, which concurrently eliminated the background image by separating the foreground signature from its surroundings. To standardise the size and placement, the signature images were then subjected to size normalisation using the moment normalisation method.

The authors' innovation is found in their feature extraction section, where they designed a Mutual Signature DenseNet (MSDN) that combines high- and low-level features and uses the local region as input to generate more unique feature representations. DenseNet-36 served as the backbone of the model, with two CNN branches that had similar structures and parameters. To fully use the properties of two signature images, they employed spatial pyramid pooling (SPP) and squeeze-and-excitation blocks (SE-Blocks). The similarity metric for a local region is computed using a Siamese network with MSDN, but the inter-signature similarity is computed by summing values of several local regions.

Wsing False Reject Rate (FRR), False Accept Rate (FAR), and Equal Error Rate (EER) as assessment measures, the authors tested the model on many benchmark datasets, including CEDAR, GPDS Synthetic, and CHnSig. Both the writer-dependent (WD) and writer-independent (WI) techniques were used by the authors.

Before exploring different combinations of local regions, the authors first apply their model to each region in their writer-independent experimentations. The central region with edge areas consistently outperformed other combinations involving additional regions, according to the authors' findings. In addition to experimenting with different

local region combinations, MSDN configuration is important. Using only multi-level feature representation ran the risk of overfitting, so utilising SE–Blocks improved performance on larger datasets such as GPDS and ChnSig. On bigger datasets, the combination of SE-Blocks with multi-level features produced the greatest results; however, it did not work on simpler datasets, such as CEDAR, since a smaller sample size typically necessitates a simpler model to avoid overfitting.

The writer-dependent (WD) strategy produced lower error rates across datasets compared to writer-independent (WI) approach. WD outperformed even with just one reference writer, and adding more references further reduces errors; 12 references produced the lowest rates.

Finally, the suggested method fared better in WI on both CEDAR and GPDS datasets. Although it only used threshold modification using test writer reference samples, it also performed better in WD settings. Unlike other approaches that retrain their models directly on the test writer's data, no network model fine-tuning was done.

2.2 Loss functions

2.2.1 Hard Negative Examples are Hard, but Useful

A triplet loss function, which uses a reference signature as the anchor image, an original signature as the positive image, and a forged signature as the negative image, is frequently used in signature verification systems. However, optimising triplet loss in deep metric learning is found to be challenging [2]; choosing the hardest negative examples for deep metric learning causes bad local minima in the early phases of the optimisation. To prevent this, authors have developed semi-hard triplet mining, whereby the negative examples are almost as close to the anchor as positive examples.

[2] found that if the gradient of the loss function does not account for the normalisation of feature vectors onto a hypersphere during backpropagation, much of the gradient information gets discarded when the features are re-normalised, especially for triplets with closely located points. Furthermore, the loss gradient may inadvertently push these points closer together rather than farther apart, contradicting the intended goal.

The authors developed Selectively Contrastive Triplet Loss (SCT), which eliminates anchor-positive pairs and concentrates only on anchor-negative pairs, in order to overcome these difficulties (Equation 2.2.1.1). SCT effectively improves the ability to discern closely comparable but semantically unrelated inputs by applying a contrastive loss that penalises high similarity between dissimilar samples.

$$L_{SC}(S_{ap}, S_{an}) = \begin{cases} \lambda S_{an,} & \text{if } S_{an} > S_{ap} \\ L(S_{ap}, S_{an}), & \text{others} \end{cases}$$

The similarity score between the anchor and a positive sample is denoted by in Equation 2.1.1, whereas the similarity score between the anchor positive and a negative sample is denoted as . When the negative closer to the anchor than the positive , the scaling factor, , is applied, penalising such occurrences more severely.

Using the CUB200 (CUB), CAR196 (CAR), Stanford Online Products (SOP), In-shop Cloth (In-shop), and Hotels-50K (Hotel) datasets, the authors assessed their approach compared to hard negative (HN) and semi hard negative (SHN) mining strategies. The images are augmented with typical data augmentations and normalised

with channel means and standard deviations. Stochastic gradient descent (SGD) was used to train all models without the use of momentum. SCT showed better capacity to avoid convergence to bad local minima than HN and SHN techniques. By only reducing anchor-negative similarity rather than enforcing distance between anchor-positive pairs, the model promotes a more dispersed embedding space, reducing risk of overfitting and enables more effective feature transfer across domains.

2.2.2 FaceNet: A Unified Embedding for Face Recognition and Clustering

Although there is no direct relationship between offline signature verification problems and [10], [10] still provided a valuable tool that is still used today. Facial recognition shares the same characteristics as offline signature verification problems in that both tasks require learning discriminative embeddings to distinguish between genuine and impostor samples. The triplet loss introduced in [10] became the cornerstone of deep metric learning, encouraging intra-class compactness and inter-class separation.

Specifically, [10] introduced the concept of pulling similar images (anchor and positive images) closer together while pushing dissimilar images (negative images) farther apart. This is achieved by ensuring that the anchor is closer to the positive than to the negative by at least a margin, (Figure 2.2.2.1).

$$D_{ap}+\alpha<D_{an}$$

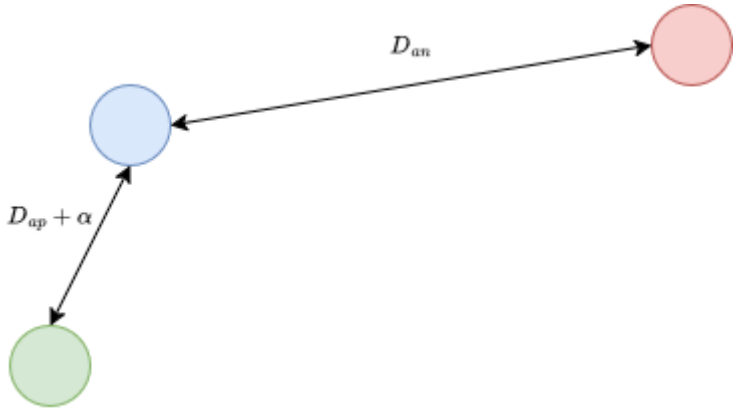


Figure 2.2.2.1 Ideal Triplet

However, many triplets generated during training may already satisfy this constraint, meaning they do not contribute to the loss and can slow convergence if still passed through the network. To address this, the authors introduced online triplet mining to dynamically compute the triplets as training progresses and specifically search for

hard or semi-hard triplets. They achieve this by constructing large mini-batches that contain multiple examples per identity, selecting anchor-positive pairs from within the batch, and then dynamically identifying negatives.

- For hard triplets, the negative is chosen as the closest impostor to the anchor (Figure 2.2.2.2).
- For semi-hard triplets, the negative is farther than the positive but still lies within the margin (Figure 2.2.2.3).

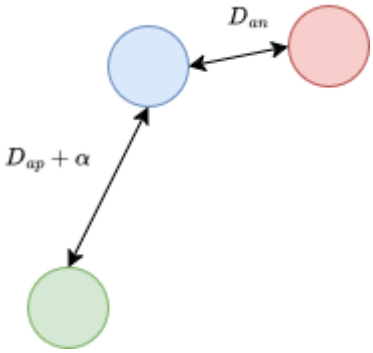


Figure 2.2.2.2 Hard Triplet

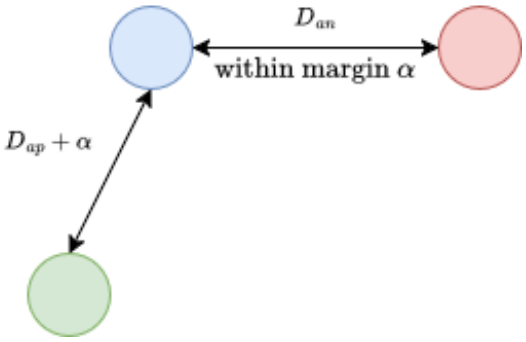


Figure 2.2.2.3 Semi Hard Triplet

However, in practice, relying solely on hard triplets often lead to poor local minima early in training, resulting in a collapsed model. In contrast, the use of semi-hard negatives, with occasional hard negatives, provided more stable training dynamics, faster convergence, and ensured that the model continued to learn challenging distinctions.

2.3 Convolutional Neural Network (CNN) Models

2.3.1 EfficientNetV2: Smaller Models and Faster Training

[11] introduced EfficientNetV2, an improved successor to the EfficientNetV1. It trains 4x faster than prior models (EfficientNetV1, BoTNeT and NFNet) at the time and it is 6.8x smaller in parameter size. EfficientNetV2 improved upon EfficientNetV1 by altering the model's architecture; instead of utilising only MBConv, the authors used both MBConv and Fused-MBConv in the early layers, resulting in faster training speed with minimal overhead on parameters and FLOPs.

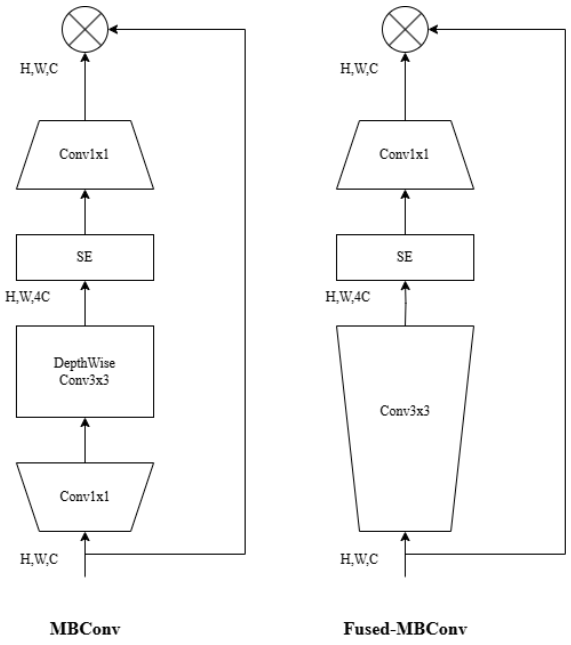


Figure 2.3.1.1 Structure of MBConv and Fused-MBConv [11]

Additionally, EfficientNetV2 also utilises smaller expansion ratio for MBConv as it has less memory access overhead, improving its efficiency even further. Smaller 3x3 kernel sizes are used, but with more layers to make up for the reduced receptive field. Finally, this model removes the last stride-1 stage found in the original EfficientNetV1; EfficientNetV1 needed that because of its larger parameter size and memory access overhead, which is no longer an issue with EfficientNetV2.

EfficientNetV1 is slower when very large images are used and depthwise convolutions are also slow in early layers. It was found that EfficientNetV1 large image size causes high memory usage, and the model had to be trained with smaller batch size due to the limit of total memory on GPU/TPU.

When trained and tested on ImageNet, EfficientNetV2 outperformed other models (ConvNets and Transformers) with better accuracy and parameter efficiency. One notable result is that EfficienNetV2 achieved comparable results to EfficientNet-B7 (largest model in the EfficienNetV1 family) while training 11x faster with the same resources. This model also outperformed other new models at the time such as RegNet and ResNet.

For a closer comparison between EfficientNetV2 and EfficientNetV1, the authors trained both models similarly. It was found that the mid-sized model EfficientNetV2-M still largely outperformed EfficientNets: EfficientNetV2-M were 4.1x faster in training and 3.1x faster in inference while having 17% lower parameters and 37% reduction in FLOPs. These improvements are attributed to the model's architecture. The authors also compared small-sized models from both families. They found that the EfficientNetV2 models were still faster and had greater parameter efficiency.

2.4 Limitations of Previous Studies

Most of the existing studies agree that deep learning approaches generally struggle with the lack of diverse training data. For example, authors in [1] pointed out that real-world scenarios often do not provide large volumes of clean signature images, causing a lack of training data. The lack of training data may cause deep learning models to overfit easily, resulting in subpar models.

Moreover, there exists an imbalance between the distribution of genuine and forged signature images, which can affect the performance of machine learning models [8]; in real life, the number of forged signatures far outnumbers the count of genuine signatures, limiting the amount of data to train the model. Additionally, there is substantial intra-class variation in signatures; according to [12], the probability of two signatures being identical is extremely low. Higher rates of false positives and false negatives for deep learning-based verification systems are among the difficulties caused by this variability. According to [13], deep learning models struggle with intra-class variations that may be obvious to human beings; however, performance can be improved by reducing intra-class variance.

Additionally, offline signatures inherently lose most of the dynamic information – writing speed, pen pressure, tilt, and stroke order and direction [1]. This is because offline signature images are acquired by scanning documents. The challenge of

inadequate data is made worse by the loss of information, which restricts the quantity of information that researchers may work on.

In terms of optimisation, works that involve triplet loss face challenges in optimising with the hardest negative examples (for example, skilled forgeries) often lead to bad local minima in early phases [10]. Although, the use of semi-hard triplets alleviates this issue, it may still cause issues brought up by [2], where poor triplet loss optimisation may pull the negative points closer to the positive and anchor points instead of separating them. Furthermore, triplet loss relies on large mini-batches with many positives per batch, as shown in [10]. This is not feasible for signatures, as signature datasets are much smaller, leading to poor triplet diversity and weaker optimisation. The conventional triplet loss also assumes positives are relatively consistent, but that is not the case for signatures; it may lead to separating genuine samples apart rather than clustering them.

2.5 Proposed Solutions

The proposed model utilises an alternative loss function that is adopted from [2], which has yet to be explored in the domain of offline signature verification. The original design proposed by [2] complements the challenges of offline signature verification well. It can help improve optimisation without compromising recognition and filtering of skilled forgeries.

The model will also make use of EfficientNetV2 [11] as the backbone of the model to make use of its lighter, yet faster convergence speed.

The main driving force behind L_{SC} lies in its capacity to increase discriminative power while improving compute efficiency. Faster convergence and more robust generalisation in situations with substantial intra-class variability can be attained by concentrating solely on aggressively pushing hard negatives away. Additionally, the loss function L_{SC} reduces the number of pairs or triplets that are computed at a time, primarily due to its behaviour of ignoring easy negatives and lower emphasis on optimising anchor-positive pairs.

Additionally, to ensure that this loss function converts well into this domain, it will be modified to include positive distance enforcement. The original design may be

sufficient for easier or general datasets, but the unique characteristic of this domain demands a stricter approach.

To make up for the challenges of imbalanced dataset, this project will alter the PK sampling into $PKFM$. This sampling method provides control over the intra-class and inter-class signatures that is fed to the model, leading to more balanced batches during training.

CHAPTER 3 SYSTEM MODEL

In this section, information on the proposed model, including the loss function and its behaviour under various situations, is provided. This section covers training, evaluation, and testing methodologies, as well as the project completion timeline.

3.1 Model Overview

The proposed system follows a deep metric learning pipeline, as shown in Figure 3.1.1. Signature images are first processed before being passed into an EfficientNetV2-M backbone. The first stage of preprocessing is designed to binarise images, removing paper artefacts and ensuring only the strokes of the signatures remain. In this stage, CLAHE, Gaussian blurring, and binarisation are applied. During the training of the model, data augmentations (resizing, rotating, scaling, and random cropping) are applied to introduce variability and improve generalisation. The extracted features are projected into an embedding space, L2-normalised, and compared using cosine similarity. Verification is performed by comparing the similarity score via a threshold-swept evaluation. The best performing threshold is chosen based on the lowest Equal Error Rate (EER) yielded. In addition to quantitative evaluation, Grad-CAM is applied to inspect the image regions that influence the model's decision (Figure 3.1.2).

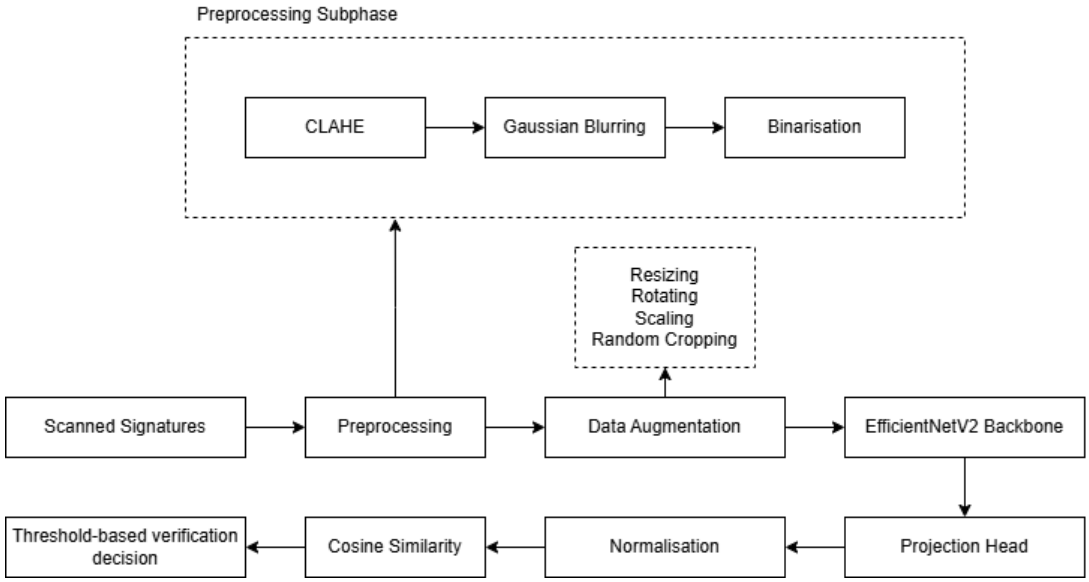


Figure 3.1.1 Model Overview

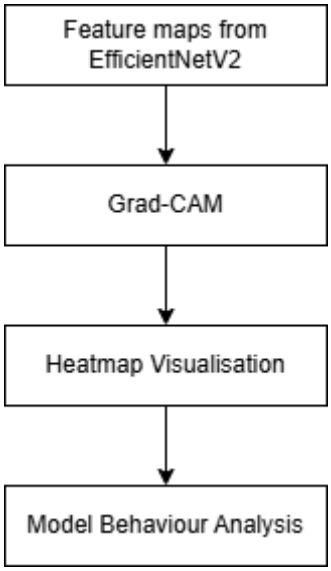


Figure 3.1.2 Grad-CAM Analysis Overview

3.2 Feature Extraction Backbone

EfficientNetV2-M was used as the feature extraction backbone. The original classification head was removed because the task is not a classification task. Instead, the backbone output was passed into a projection head to generate a fixed-dimensional embedding. This allowed the model to compare unseen signers through similarity scoring rather than predicting a fixed identity class. Additionally, the input layer of the backbone was modified to accept grey-scale images rather than RGB images.

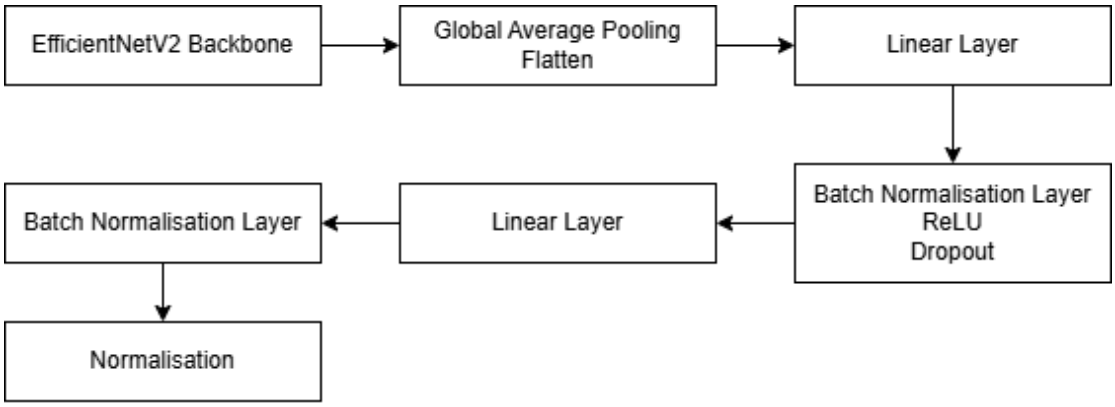


Figure 3.2.1 Model Architecture

3.3 Loss Function

A triplet loss function is trained using sets of three images (x_a, x_p, x_n). Here, x_a represents the anchor image. The x_p is a positive image from the same class as the anchor. The x_n is a negative image from a different class. Cosine similarity, bounded by $[-1, 1]$, is used to measure the similarity between anchor-positive and anchor-negative pairs embedded on a hypersphere [2]. Each pair is represented as such:

$$S_{ap} = f_a^T f_p$$

$$S_{an} = f_a^T f_n$$

The L_{SC} function proposed by [2] mitigates the challenge of poor optimisation during gradient calculation and minimises gradient entanglement that can collapse the embedding space caused by hard negative mining in triplet loss. The modification that ignores anchor-positive pairs and easy negatives, thereby focusing on penalising hard negatives, is beneficial for the domain of offline signature verification. When a forged signature is closer to the genuine signature than another genuine signature, the loss decouples and focuses entirely on pushing the forgery away. By placing a heavier emphasis and penalty on difficult forgeries, the model can be trained to be more robust towards skilled forgeries. Decoupling the anchor-positive pair from the anchor-negative pair helps prevent gradient entanglement, thereby avoiding the counter-intuitive behaviour of pulling instead of pushing when shifting samples on the hypersphere.

Furthermore, the design of L_{SC} reduces overfitting in models with its inherent logic of ignoring anchor-positive pair optimisation during training. This leads to looser intra-class clusters, helping the model generalise to new, unseen angles and lighting conditions for objects from the same class [2]. This design can be beneficial for generic or easy datasets such as CAR196 and CUB200, both of which were used in [2] to train and test the model; however, in the domain of offline signature verification, this design may backfire. This can be attributed to the high intra-class variance present in this domain. If no modification is made, the model may not be trained sufficiently to keep genuine signatures together. Because genuine signatures may drift apart, if the same signer were to leave his signature at a separate time, this new sample may fall outside of the decision boundary, leading to a false negative.

To attenuate this potential vulnerability, an explicit **Positive Pull Regularisation** (μL_{pull}) was added. This regularisation enforces a constant constraint on genuine pairs, ensuring that signatures from the same user remain clustered around a specific margin while ensuring that the model aggressively separates skilled forgeries, providing a balance between pulling and pushing.

With the addition, the complete loss function L_{SC+} formula is defined as:

$$L_{SC+}=L_{SC}(S_{ap},S_{an})+\mu L_{\text{pull}}$$

Where:

- L_{SC} is Selectively Contrastive Triplet Loss
- L_{pull} is Positive Pull Regularisation
- μ is a hyperparameter weighing the positive pull

The L_{SC} handles the inter-class separation, whereby it switches behaviour according to the difficulty of anchor-negative pairs [2].

$$L_{SC}(S_{ap},S_{an})=\begin{cases}\lambda S_{an}, & \text{if } S_{an}>S_{ap}\\ L(S_{ap},S_{an}), & \text{otherwise}\end{cases}$$

Where:

- $S_{ap}=f_a^T f_p$ is cosine similarity between a pair of anchor and positive signatures
- $S_{an}=f_a^T f_n$ is cosine similarity between a pair of anchor and negative signatures
- λ is weighting factor for hard negatives

Compared to the standard triplet loss implementation, this approach is more computationally efficient. When a negative sample is positioned closer to the anchor than a positive sample, the loss function decouples the pairs; the negative sample then receives an exponentially higher weight and is pushed aggressively away from the anchor within the hypersphere. Conversely, when a negative sample is sufficiently distant, the model incurs no penalty and simply ignores it, switching to a smooth, self-normalising log-softmax and receiving vanishingly small weights.

The L_{pull} component is defined as follows:

$$L_{\text{pull}} = \begin{cases} m\text{-}S_{ap}, & \text{if } S_{ap} < m \\ 0, & \text{if } S_{ap} \geq margin \end{cases}$$

Where:

- m is margin

The margin ensures that the model pulls the positives only up to a certain threshold, allowing it to maintain tight clusters while generalising to the natural variations in human handwriting. If a margin is omitted, the model may be encouraged to keep optimising positives, ignoring its initial purpose of maximising negative distances, which can subsequently lead to the collapse of the embedding space. Once the anchor-positive similarity exceeds the threshold, the gradient from the positive pull becomes 0, allowing the model to focus on separating negatives.

3.4 Intuition

In the proposed loss function, the model maps each signature image into a high-dimensional embedding space. For a specific signer, a reference signature serves as the anchor (A), while additional genuine samples from the same individual are designated as positives (P). All forgeries or signatures from different signers are treated as negatives (N). The loss function is engineered to structure this space so that genuine signatures form a cohesive cluster, while negatives are aggressively pushed apart. During optimisation, the loss function prioritises pushing hard negatives from A, focusing on samples that lie within the decision boundary to refine the model's discriminative power rather than expending computational effort on easy negatives. Concurrently, an explicit positive-distance enforcement term pulls P toward A. This pulling force is constrained by a predefined margin, preventing overfitting and preserving the model's ability to generalise across the natural intra-class variations of a human signature. Consequently, A resides at the centroid of a writer-specific cluster where positives are drawn inward and hard negatives are repelled outward, progressively sharpening the decision boundary between genuine and forgery.

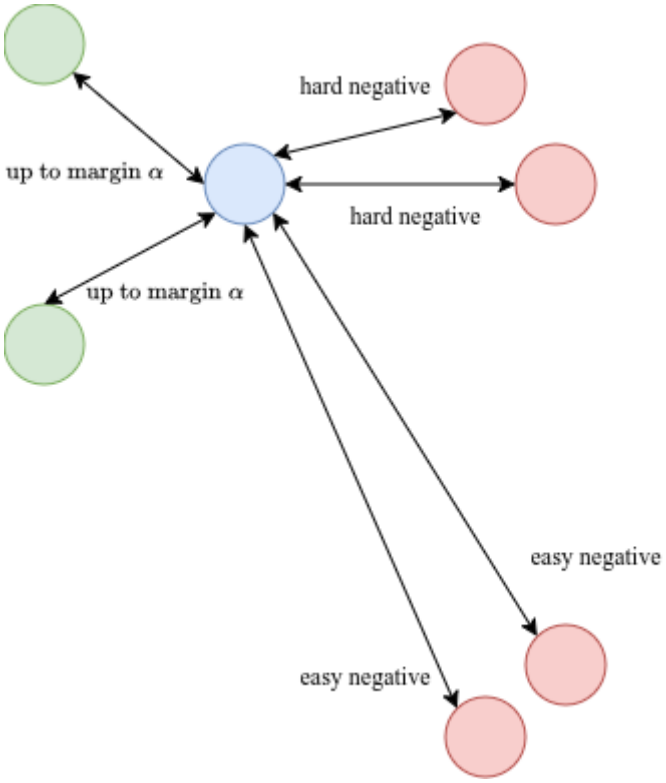


Figure 3.4.1 Embedding Space Dynamics Under L_{SC+} Loss Function

To illustrate the practical application of the proposed writer-independent model, this subsection presents three representative scenarios using a bank cheque-clearing workflow. For each customer, a small enrolment set of genuine signatures is processed by the EfficientNetV2-L_{SC+} backbone to generate reference embeddings, which establish a writer-specific cluster. During the verification phase, the signature from an incoming cheque is pre-processed and mapped into the embedding space as a query vector (q). The system then calculates the distance between q and the stored genuine embeddings (P) of the claimed account holder, applying a threshold-based decision to determine authenticity.

Scenario 1: Easy Forgeries (Inter-class Negatives)

Assume that Customer A has three reference signatures enrolled in the system. An imposter produces a crude imitation that is visually distinct from the enrolled samples. When this forgery is processed, the model ensures that such clearly dissimilar negatives are mapped far from the compact cluster of genuine signatures. The cosine similarity between reference embeddings and the forgery embedding is therefore low, causing the aggregated score to fall well below the acceptance threshold. The model classifies the signature as a non-match with high confidence, allowing the item to be automatically rejected or flagged for review, depending on the bank's operating policy.

Additionally, this scenario applies when a signature from a different customer is compared to Customer A's list of reference signatures. Moreover, signatures from different times by Customer A that are visually different will also be flagged as a forgery.

All in all, this model flags drastically visually distinct signatures as forgery, regardless of whether they are from genuine signers.

Scenario 2: Skilled Forgeries (Intra-class Hard Negatives)

In a more sophisticated attack, a skilled forger presents a forgery that closely mimics the visual characteristics of the genuine signature. These signatures are classified as hard negatives and are aggressively pushed from the genuine cluster by the loss function during training. At deployment, the embedding of this skilled forgery may be closer to the genuine cluster than that of a random forgery. Consequently, the similarity between the skilled forgery and the reference signature may be below the acceptance threshold or within a narrow, uncertain range. The system flags the cheque as suspicious, prompting a bank staff member to review the cheque manually.

The implementation of a deep-learning based offline signature verification drastically reduces the volume of documents that bank staff have to verify, allowing them to prioritise high-risk, hard-to-spot cases.

Scenario 3: Natural Variations (Intra-class Positives)

This scenario extends the logic of the aforementioned scenario one. Consider a genuine signature from Customer A produced at a different time, exhibiting inherent variations in stroke thickness or spatial proportions. Numerous factors influence these strokes, ranging from mechanical variables such as pen thickness and ink viscosity to the signer's physiological and psychological state. In a conventional triplet-loss system, such outliers are at risk of being mapped outside of the cluster, leading to misclassification (False Rejection, or Type I error), as the standard triplet-loss-trained model can be overly punishing toward intra-class variance.

In the proposed model, the explicit positive-distance enforcement term penalises genuine pairs that are mapped too far apart, yet it remains lenient enough to allow for a naturally sparse cluster. This approach encourages a cohesive representation of the writer's underlying style. Consequently, the feature vector for this new signature remains within the acceptance margin of the genuine cluster, yielding a similarity score above the threshold. Nonetheless, this scenario is contingent on the signature not being drastically different from the stored references, as extreme deviations may still fall outside of the model's learnt tolerance.

3.5 Methodologies and General Work Procedures

The lifecycle of this project followed the CRISP-DM methodology. Originally meant for data mining projects, this methodology has been adopted in many deep learning research and development (R&D) projects. This is a flexible methodology that allows developers to shift between checkpoints and ensure the defined objectives are closely followed. This can help minimise scope creep and under-deliverance of promises as the R&D progresses.

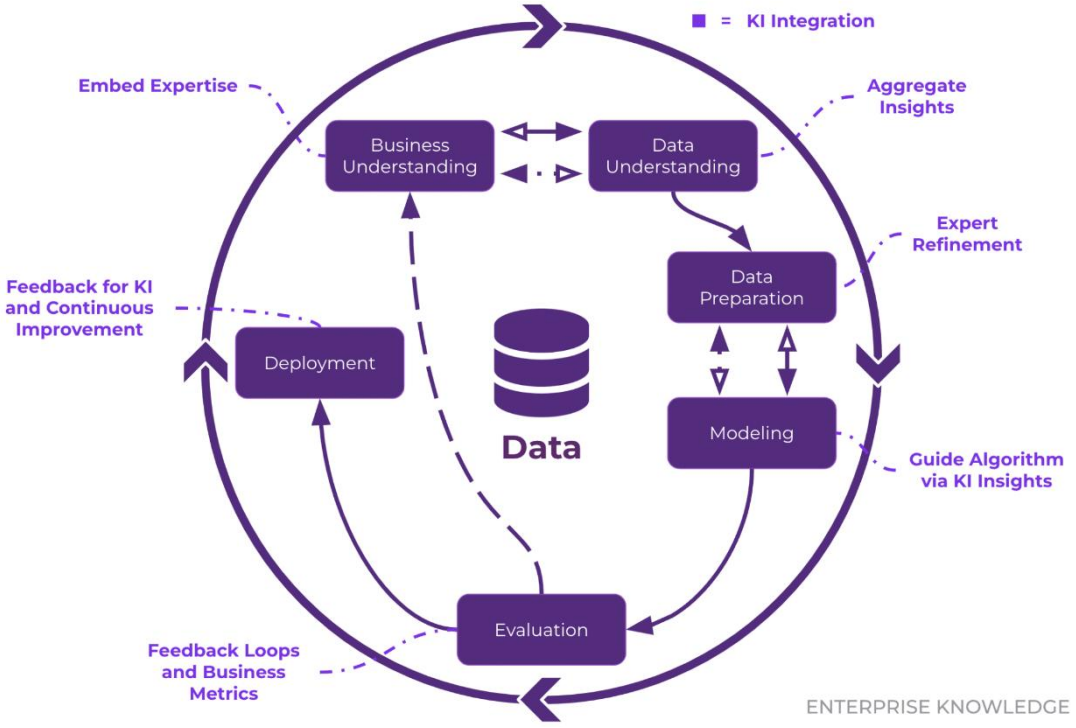


Figure 3.5.1 CRISP-DM Methodology

Based on CRISP-DM:

Table 3.5.1 Activities to be Completed

Stages	Activities
Business Understanding	Defining the problems and limitations faced in offline signature verification problems. Establish project objectives
Data Understanding	Research and collect signature dataset(s) Perform dataset exploratory analysis to gain information Identify dataset limitations
Data Preparation	Pre-process signature images Partition datasets into training validation and testing sets
Modelling	Perform transfer learning on an existing CNN model Implement the proposed L_{SC+} loss function Train and fine-tune model
Evaluation	Utilise metrics such as accuracy, Equal Error Rate (EER), False Acceptance Rate (FAR), False Rejection Rate (FRR), and Area Under the Curve (AUC) Compare results of the model trained with L_{SC+} against vanilla triplet loss implementation
Deployment	Report the result of the model utilising L_{SC+}

3.6 Timeline and Gantt Chart

Figure 3.6.1 Project I Gantt Chart

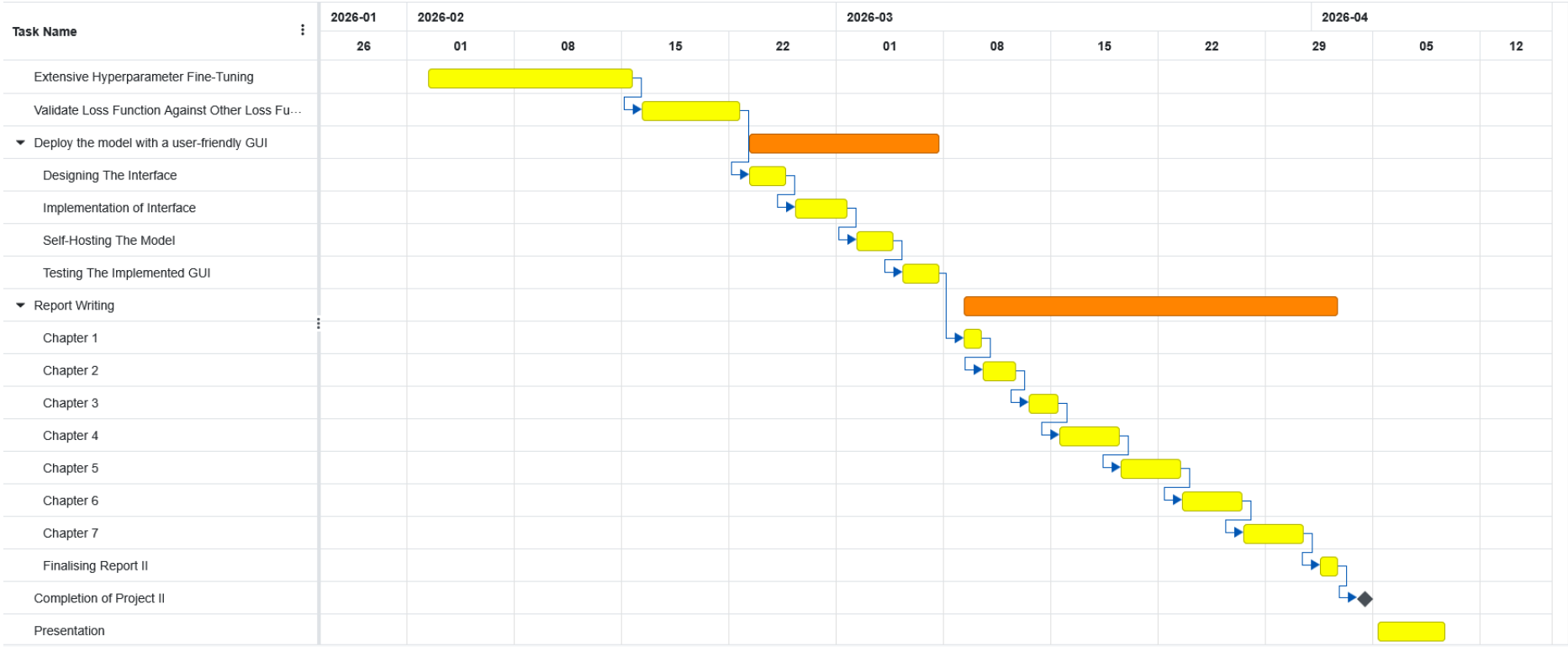


Figure 3.6.2 Project II Gantt Chart

CHAPTER 4 SYSTEM DESIGN

4.1 System Overview

The workflow comprised two major phases: the data collection and preparation phase, and the training and testing phase.

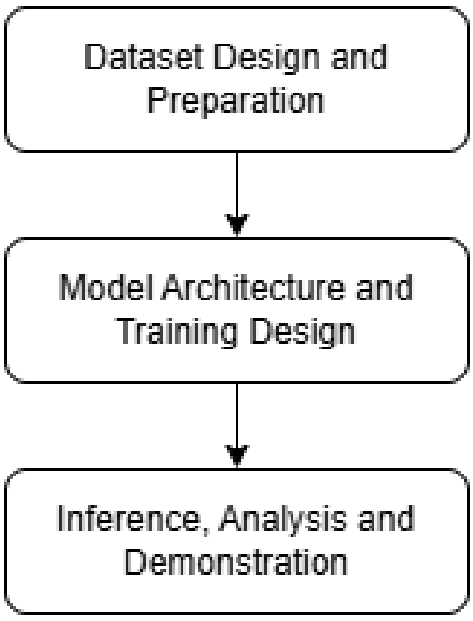


Figure 4.1.1 Overall System Design Pipeline

This chapter presents the design of the proposed system. To provide a clear overview before discussing individual components, the overall system design is first introduced. The pipeline begins with signature image collection and preprocessing, followed by signer-level dataset splitting, mini-batch construction using PKFM sampling, metric learning with an EfficientNetV2-based embedding model, threshold-based evaluation, and Grad-CAM analysis for model interpretability

4.2 Dataset Design and Preparation

4.2.1 CEDAR Dataset

The dataset comprises 55 signers, each with 24 genuine and 24 forgeries. Each image is preprocessed using Otsu's method to remove background noise and make the signature strokes stand out. To compensate for the low volume of natural handwritten offline signature images, data augmentation, such as resizing (384, 384), rotating ($\pm$ 5), scaling (0.9-1.05x), and random cropping, was applied to the dataset. These augmentations not only increased the number of signature samples for training but

also increased sample diversity and simulated realistic variations in pen pressure, orientation, and scanning conditions (off-centred images). By introducing variations in orientation and spatial positioning, the model was forced to learn pose-invariant features, ensuring that slight misalignments or scale differences in a scanned cheque do not adversely affect the resulting embedding. This resulted in a model that can generalise to real-world signatures, which are often captured imperfectly.

According to [11], a smaller image sizes result in lower network capacity and does not require strong regularisation. Similarly, this is the case with offline handwritten signature images; extremely high resolutions are generally not required. [11] showed that weak augmentations achieve the best accuracy when image size is small, which is also often the case for cropped signature images. Such constraints are to reduce noise, ensuring the model only learns crucial, defining features.

Figures 4.2.1.1 to 4.2.1.4 shows examples of CEDAR signature samples:

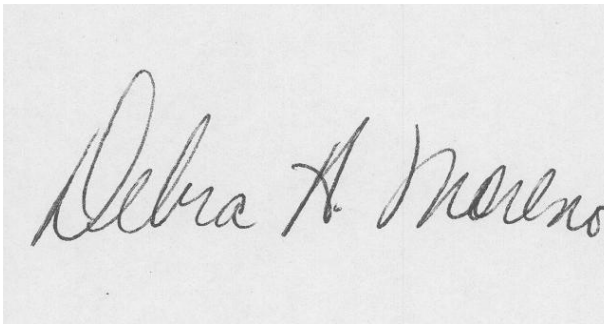


Figure 4.2.1.1 CEDAR Genuine Signature (9-1)

Figure 4.2.1.2 CEDAR Genuine Signature (25-11)

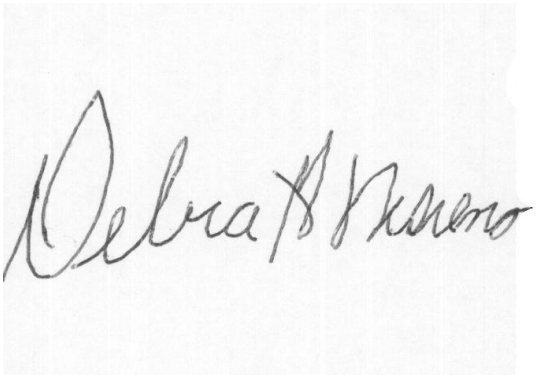


Figure 4.2.1.3 CEDAR Forged Signature (9-11)

Figure 4.2.1.4 CEDAR Forged Signature (25-5)

The dataset was split by signer IDs rather than by images (Table 4.2.1.1), ensuring that no signature from a given signer appeared in both training and test sets. This prevented leakage of test images into the training set, enforcing subject-independent evaluation and mitigating overfitting. This restriction was necessary to ensure that the model's performance on the test set and its generalisability was genuine.

Table 4.2.1.1 Dataset Splitting

Split	Writers	Genuine Images	Skilled Images	Forgery	Total Images	Signer Overlap
Training	38	912	912		1824	No
Validation	6	144	144		288	No
Testing	11	264	264		528	No

The dataset can be found at: https://cedar.buffalo.edu/signature

4.2.2 Self-Sampled Dataset

An additional, smaller self-sampled signature dataset was collected to evaluate the model's robustness towards signature images captured with modern devices, contrasting the more dated, albeit clean dataset used during training. Each participant contributed four genuine signatures and four forgeries of another individual. To simulate real-world attacks as closely as possible, participants were allowed to practice their forgeries before the final recording committing, creating a difficult dataset designed to stress-test the model.

Each signature was signed in a card-sized box; this provides each signer with sufficiently large and consistent space. This spatial constraint was necessary to ensure that the signers were able to sign without the subconscious cramping that often occur in smaller, restrictive forms. Additionally, this approach created a grid that made scanning each sample efficient; scanning and cropping each signature were streamlined and consistent.

Furthermore, the same preprocessing and data augmentations pipeline (CLAHE, Otsu thresholding, resizing, and cropping) applied to the test set were applied to the self-sampled signatures; this standardised the input distribution between synthetic and real-world samples. This allowed the evaluation to be more indicative of the model's

actual capabilities rather than being skewed by distribution shift. If the input distribution is too far apart, the model's inherent embedding space will not align, leading to a drop in performance, regardless of training quality.

For example, signatures in CEDAR dataset are tightly cropped to the strokes, forcing the models to prioritise local spatial features rather than global representation. If the self-sample signatures contain excessive whitespace due to improper cropping, these crucial local representations may be obscured, causing the model to extract sub-optimal feature embeddings. Ultimately, this results in higher False Positive Rates (FPR).

Figures 4.2.2.1 to 4.2.2.4 shows examples of CEDAR signature samples:

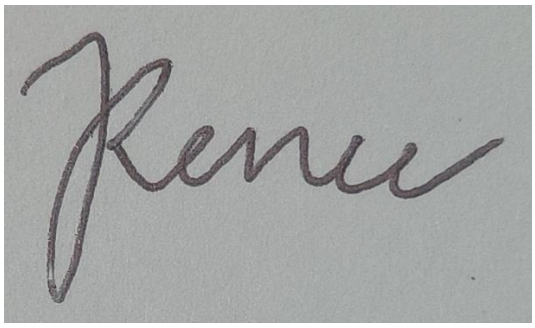


Figure 4.2.2.1 Self-Sampled Genuine Signature (2-1)

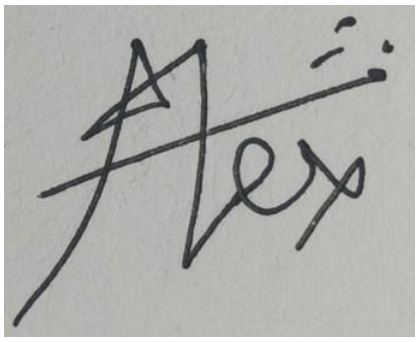


Figure 4.2.2.2 Self-Sampled Genuine Signature (5-1)

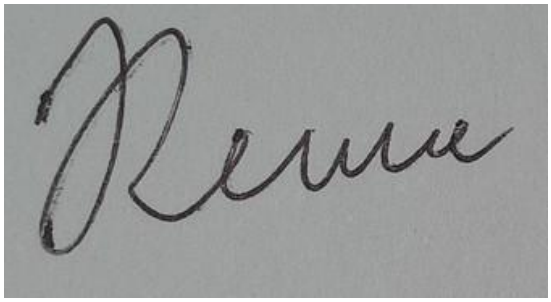


Figure 4.2.2.3 Self-Sampled Forged Signature (2-2)

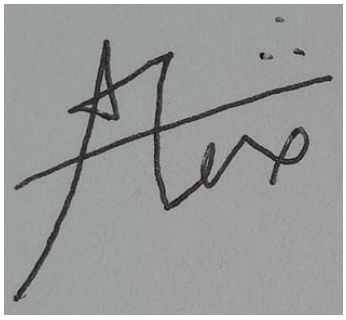


Figure 4.2.2.4 Self-Sampled Forged Signature (5-1)

Remaining examples of these signature images are available in Appendix A.

4.2.3 Preprocessing of Signature Images

This workflow began with the collection of publicly available offline signature dataset (CEDAR). These signature images, reflecting real-world scenarios, were scanned in grayscale as colour information was generally considered unnecessary for biometric

extraction. To account for potential edge cases, safety measures such as image format checking and colour space conversion were implemented.

Image format checking ensured that only valid images entered the pipeline, which prevented compatibility issues. Colour conversion was required because the model accepted only single-channel input; although the source images were grayscale, they were often stored in a 3-channel RGB format with identical values across all channels. Standardising each image to a single channel prevented interruptions during training and optimised memory usage.

Moreover, while greyscale was utilised, it was observed that some samples exhibited poor contrast, which rendered the ink strokes difficult to discern from the background. This lack of clarity directly impacted the model's feature extraction performance, as the network struggled to differentiate subtle stroke textures from background noise, such as paper artefacts. Figure 4.2.3.1 shows an example of signature images before preprocessing

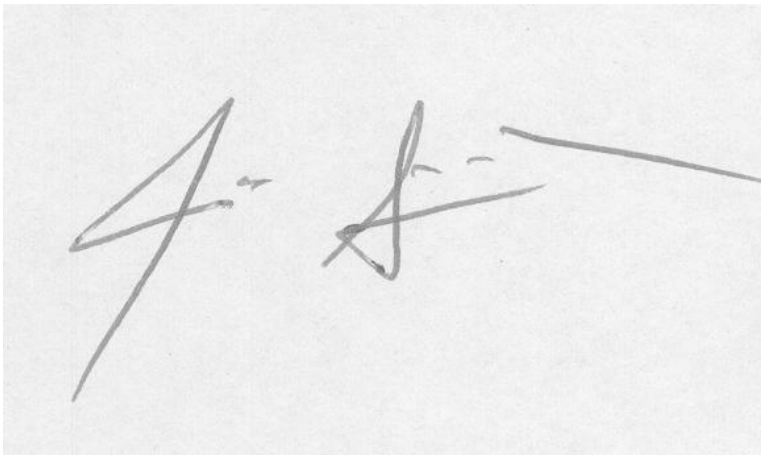


Figure 4.2.3.1 Example CEDAR Scan Before Preprocessing

To ensure the strokes were prominently isolated, each image underwent a series of preprocessing steps, beginning with contrast enhancement. This step was designed to improve the contrast between the strokes and the image background. However, this presented a side effect: worsening of paper artefacts in the images.

To mitigate this, Gaussian Blurring was applied to the images. This technique served to smooth out the noise and soften the sharpened artefacts, ensuring that the model focused on the structural integrity of the strokes rather than high-frequency background interference.

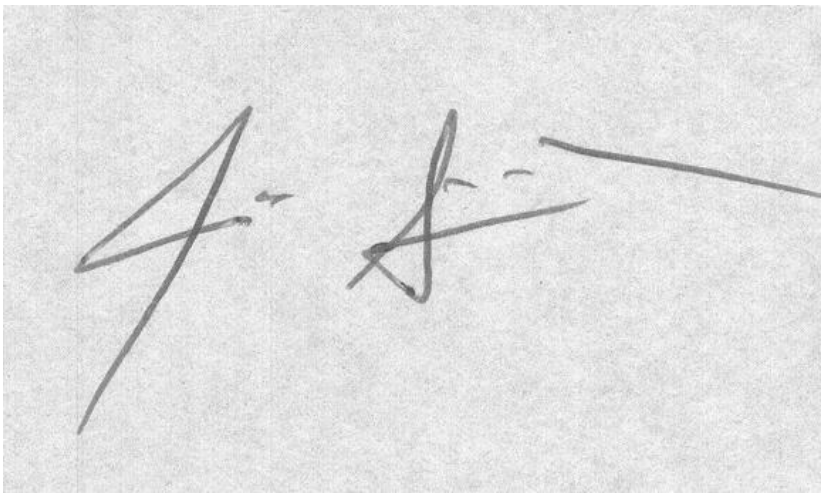


Figure 4.2.3.2 Contrast Enhanced Sample

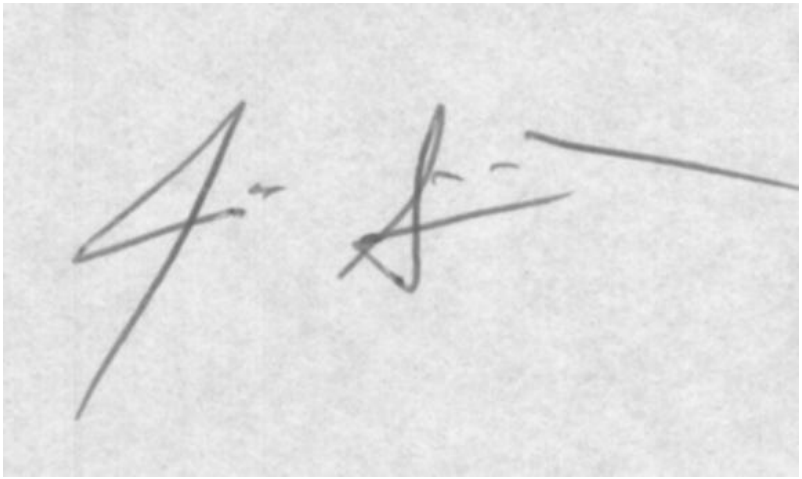


Figure 4.2.3.3 Gaussian Blurred Sample

By smoothing these artefacts and widening the intensity gap, the image histogram becomes more distinctly bimodal, allowing Otsu's Method to calculate an optimal global threshold that minimises intra-class variance.

Before

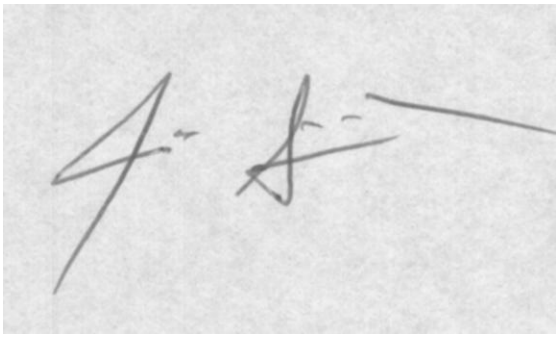


Figure 4.2.3.4 Before Otsu's Thresholding

After

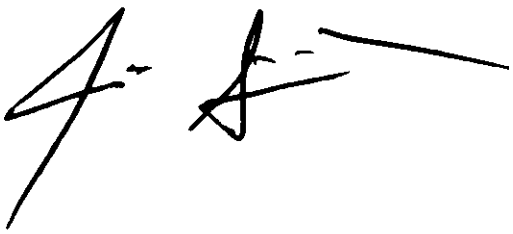


Figure 4.2.3.5 After Otsu's Thresholding

Following binarisation, the samples were restructured and renamed to ensure consistency across datasets. Each sample was grouped under its type (original or forgery) and saved under its source, such as CEDAR, as shown in Figure 4.2.3.6.

This organisation prevented the leakage of signatures between datasets, enabling a more credible training and evaluation pipeline. The samples were renamed with the following format: `[original/forgeries]_[id]_[count].`

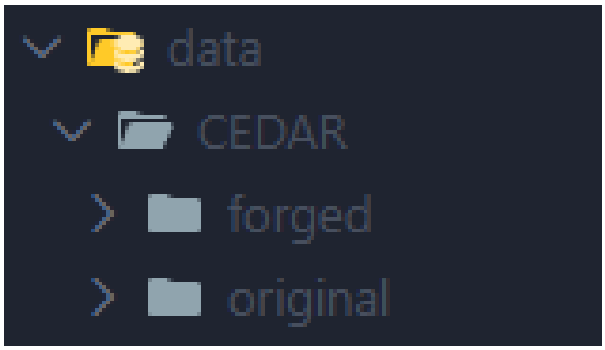


Figure 4.2.3.6 Folder Structured

The detailed flow of this sub-pipeline can be viewed in Figure 4.2.3.7.

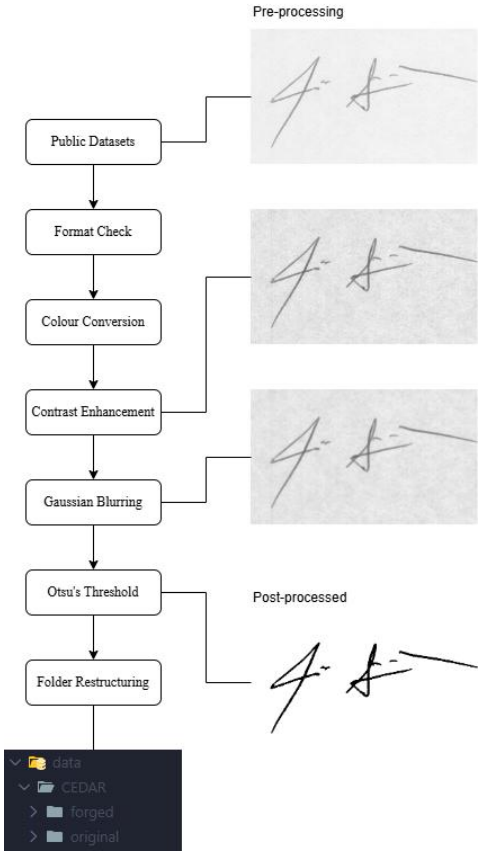


Figure 4.2.3.7 Dataset Design and Preparation Subpipeline

4.3 Sampling and Batch Construction

The model was trained using a mini-batch training strategy [16] with a sampling technique that extended the conventional PK sampling strategy, $PKFM$. In the standard PK sampling technique, a batch is formed by selecting P signers and K signatures per signer, controlling the balance between inter-class and intra-class samples. However, it does not account for the balance between hard negatives and easy negatives.

$PKFM$ introduces two additional parameters, F and M, which regulate the number of intra-class and inter-class negatives. Increasing F raises the likelihood of countering skilled forgeries (hard negatives), while increasing M introduces more signatures from different signer classes (easy negatives), thereby enriching the diversity of each batch.

The resulting batch size is given by:

$$\text{Batch Size} = P \times (K + F + M)$$

This modification ensures that each batch contains a controlled mix of genuine signatures, skilled forgeries, and easy forgeries, thereby enabling the model to effectively distinguish between broad signer classes and subtle forgery attempts, separate various classes, and also separate genuine signatures from forgeries. From an industry standpoint, offline signature datasets are often imbalanced, with some signers contributing disproportionately more samples. If such an imbalance is not handled with scrutiny, certain signers may experience a disparity in verification accuracy (e.g. capturing more genuine variations). The current design mitigates this stochastic bias by balancing the dataset at the batch level, ensuring the model learns robust features across all signer classes regardless of their sample frequencies.

The adoption of $PKFM$ sampling, mini-batch training, and a custom loss function, L_{SC}, rendered training more effective in a writer-independent, few-shot learning setting. For each training step, the $PKFM$ sampling technique first selects P signers, and for each selected signer, it draws K genuine signatures and F skilled forgeries. Then, it adds M additional signatures from different signers as easy forgeries. To illustrate, with parameters set to $P=2, K=2, F=1$ and $M=2$, each mini batch consists

of two genuine signatures and one skilled forgery for each of the two selected writers. Furthermore, for each of these writers, two additional signatures are randomly sampled from the global pool of all other signers to act as easy negatives. This results in a diverse mini batch of 10 samples, ensuring the model is simultaneously challenged by subtle intra-class differences and broad inter-class variations. This construction enriches the mini batch without any manual annotation beyond the standard writer ID and genuine/forgery labels.

4.4 Model Architecture and Training Design

Training a Convolutional Neural Network from scratch would require tens of thousands to millions of labelled examples to reach satisfactory performance. Unfortunately, offline handwritten signature datasets are often limited in size, making training from scratch impractical. However, transfer learning can be utilised to improve training speed and efficiency; EfficientNetV2 [11] was be used to complement the loss function. EfficientNetV2 was trained with a compound scaling strategy that balances width, depth, and resolution, resulting in a model with accuracy that outperformed other CNNs while being 11x faster and 6.8x smaller.

Given the nature of this project, the standard EfficientNetV2 classification head was replaced with a projection head that compresses feature vectors into a 256-dimensional embedding space. This architecture utilised a sequence of Linear Layers, Batch Normalisation, Dropout, and Rectified Linear Unit (ReLU) activations. This dimensionality was chosen to maintain high representational capacity for intricate signature details while remaining computationally efficient for large-scale training. The input layer was modified to accept single-channel images as inputs, aligning with real-world signature scans.

Among the three EfficientNetV2 variants available in PyTorch, the mid-size variant, EfficientNetV2-M, was selected for its an optimal balance between computational cost and performance. This variant outperforms EfficientNetV2-S while remaining feasible to train within the available resource constraints. In contrast, EfficientNetV2-L has more than twice as many parameters as the M variant yet yields only marginal performance gains on ImageNet [11].

Table 4.4.1 Comparison Between EfficientNetV2 Variants [11]

Models	Top-1 Accuracy (%)	Parameters
EfficientNetV2-S	83.9	22M
EfficientNetV2-M	85.1	54M
EfficientNetV2-L	85.7	120M

Additionally, according to [16], reusing all pre-trained layers is not recommended, as it may destabilise the learnt weights. Generally, lower layers are more versatile than upper layers, as the feature requirements for signature verification differ significantly from those of the model's original training task. Thus, freezing the lower convolutional blocks preserves essential low-level features, such as edges and textures, while leaving the upper blocks unfrozen allows the model to specialise its high-level representations for the signature verification domain.

Training employed a linear warm-up for the first 5 epochs, gradually increasing the learning rate from 0.0001 to 0.001. This was followed by a cosine decay schedule that smoothly reduces the rate towards a minimum of $1e\text{-}6$, stabilising early training and facilitating fine-grained convergence [17].

Optimisation was performed using $AdamW$ with β_1=0.9, β_2=0.999, and a weight decay of 0.001. Unlike the standard Adam optimiser combined with naïve L2 regularisation, AdamW decouples weight decay from the gradient update. This distinction improved generalisation and effectively prevented overfitting, particularly within the constraints of small or imbalanced datasets [16].

The model was trained over a fixed number of epochs, with each epoch processing the dataset in batches determined by the proposed $PKFM$ sampling method. The EfficientNetV2 backbone extracted feature embeddings for each signature, which were subsequently passed to L_{SC+}. At the end of each epoch, the model's performance was assessed on a validation set using a threshold-free evaluation strategy.

Rather than relying on a fixed decision threshold, the model's discriminative ability was assessed across all possible thresholds by computing the Area Under the Curve (AUC) score. A model checkpoint was saved whenever the validation result showed improvement, ensuring the selection of the most robust weights for final inference.

Each checkpoint stored:

Table 4.4.2 Checkpoint Saved State

Component	Description
Model Weights	Stores the learnt parameters of the CNN backbone and projection head. These weights define how the model extracts signature features and map them into the embedding space.
Optimiser State	Contains the internal statistics of the optimiser. Restoring this ensures training continues smoothly without losing optimisation progress.
Scheduler State	Saves the learning rate schedule progression. This prevents the model from resetting to the wrong learning rate when training resumes.
Best AUC Recorded	Tracks the highest validation AUC achieved so far. Used for early stopping and model selection to ensure the final model reflects the best-performing checkpoint.
Patience Counter	Stores the count of consecutive epochs where the model has not shown improvement. Required for early stopping logic to resume accurately after loading a checkpoint.
Current Epoch's AUC	A record of the most recent evaluation score. Helps log progress and allows the training loop to resume with full awareness of previous performance trends.

Given the possibility that the model could overfit during training, an early stopping mechanism was implemented to prevent redundant computation and mitigate overfitting. This mechanism utilised a patience counter, halting the training cycle if the validation AUC score failed to improve after a specified number of consecutive epochs.

The detailed flow of this sub-pipeline can be viewed in Figure 4.4.1.

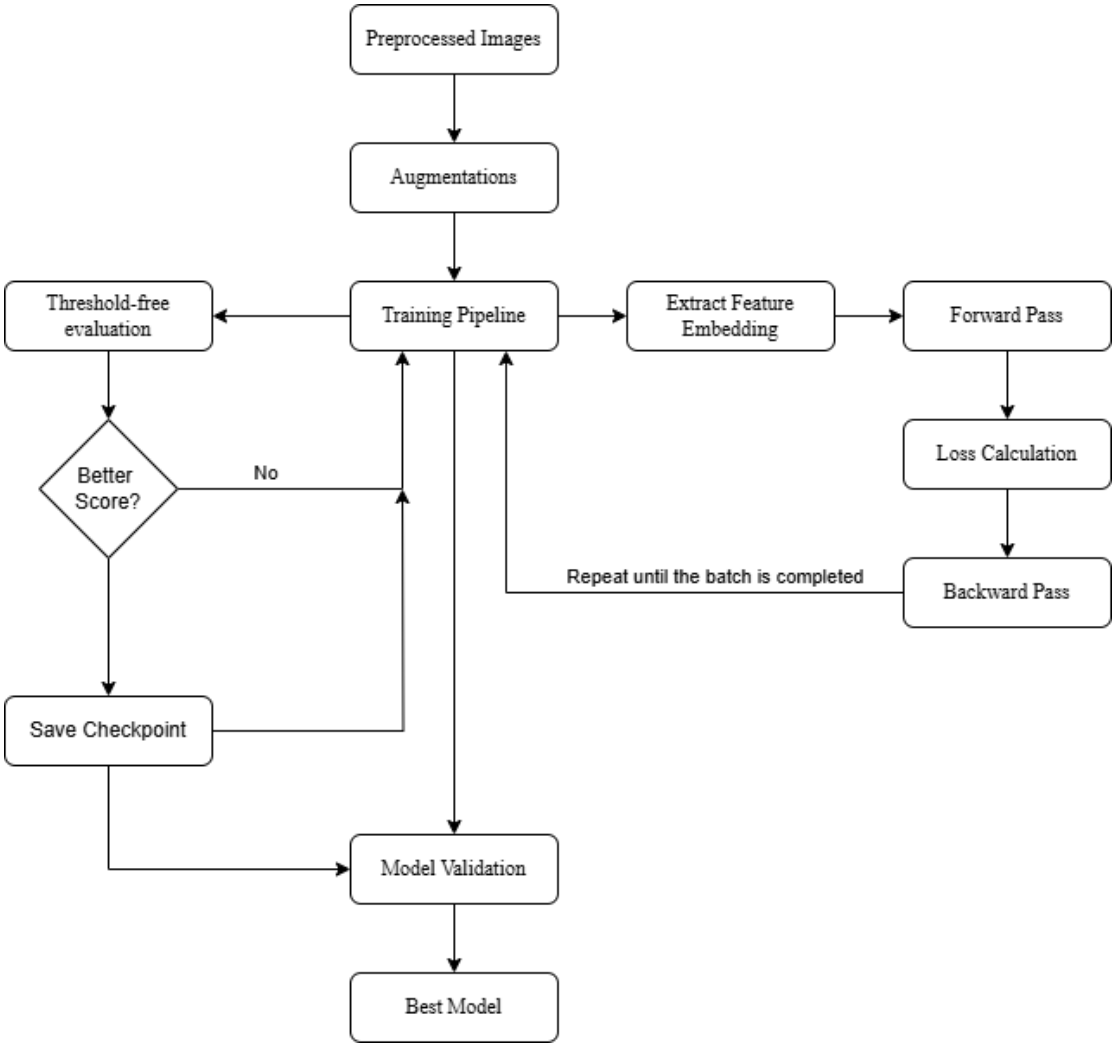


Figure 4.4.1 Training and Validation Subpipeline

4.5 Inference and Analysis

For comparison, a standard triplet loss function was implemented, supporting both hard mining and semi-hard mining. To ensure a rigorous and consistent evaluation between the conventional implementation and the custom loss function, the model architecture, learning rate scheduler, optimiser, and general training parameters remained identical across all experiments. The margin hyperparameter was fixed at 0.5 for all loss functions, ensuring that performance variations were attributable to the loss function formulation itself rather than margin selection.

Additionally, an extensive hyperparameter tuning often required hundreds of training iterations, which was computationally expensive given the resource constraints; models were trained using empirically recommended settings. Training progress for each model was visualised through line graphs, stacked area charts, and histograms. During the testing phase, model performance was quantified using the following metrics: True Positive Rate (TPR), False Positive Rate (FPR), Equal Error Rate, and accuracy.

Additionally, the average cosine similarities for anchor-positive and anchor-negative samples were monitored to analyse embedding separation. Final discriminative performance was assessed using threshold-free evaluations, specifically the Area Under the Curve (AUC) and the Equal Error Rate (EER).

Table 4.5.1 shows a summary of the models that were compared. The hard triplet mining and semi-hard triplet mining models served as the baseline models for evaluating the proposed LSC+ loss function.

Table 4.5.1 Summary of Methods

Experiment	**Backbone**	**Loss Function**	**Sampling**	**Purpose**
Baseline 1	EfficientNetV2-M	Triplet Loss + Hard Mining	PKFM	Aggressive hard-negative baseline
Baseline 2		Triplet Loss + Semi-Hard Mining	PKFM	More stable triplet baseline
Proposed		L_{SC+}	PKFM	Proposed loss-function approach

4.6 Gradient-weighted Class Activation Mapping (Grad-CAM)

To unravel the "black box" nature of the model and validate its decision-making integrity, Gradient-weighted Class Activation Mapping (Grad-CAM) was implemented; Grad-CAM produced a localisation map highlighting the regions that most strongly influenced the model's similarity score. Grad-CAM was selected because it is lightweight and compatible with CNN-based feature extractors.

Aside from validating the performance difference when training with L_{SC+} compared to the conventional triplet loss function, it was also necessary to ensure that the model was learning from the strokes of the signatures. In the context of signature verification, this served two critical diagnostic purposes:

1. **Verification Validation**
   - Using Grad-CAM, it was possible to determine whether the model was attending to high-frequency biometric features, such as stroke junctions, pen-tip pressure points, and curvature dynamics, rather than artefacts and noises.
2. **Shortcut Detection**
   - With the heatmaps generated via Grad-CAM, it was possible to identify whether the model had succumbed to learning from noises. This was observed when the activation heatmaps focused on whitespace artefacts or padding noises, which served as proxies when the model could not discern the fine differences between a genuine signature image and a highly skilled forgery.

However, it was also important to note that, while Grad-CAM offered insights into what the model was paying attention to, it did not inform us as to why the model chose those regions.

The heatmap generated for individual signature images provided insight into the model's feature extraction process. Regions with a bright red indicated areas of greater attention, while bluer hues indicated regions deemed less important by the model (Figure 5.5.2.1).

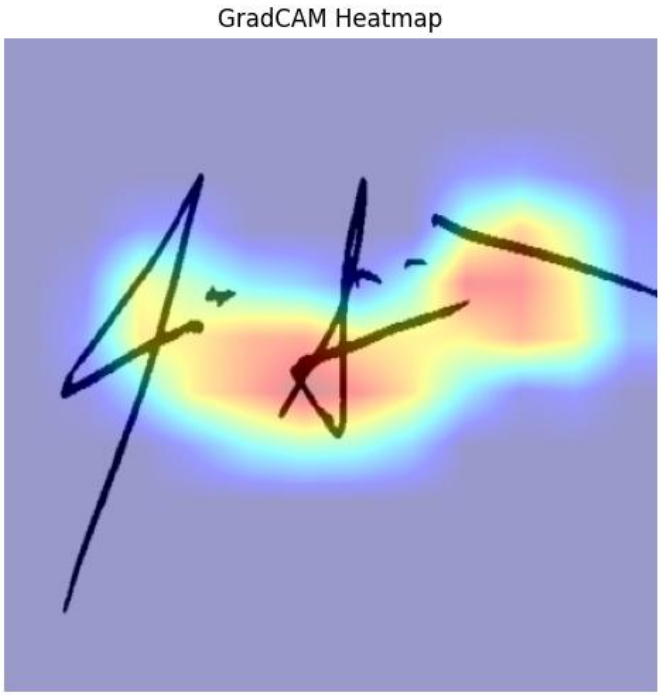


Figure 4.6.1 Grad-CAM Heatmap

The technical configuration of Grad-CAM was as shown in Table 4.6.1.

Table 4.6.1 Technical Configuration

Item	Configuration	Reasoning
Target layer	Final convolutional block of the EfficientNetV2-M backbone, implemented as `model.backbone[-1]`	Retains high-level spatial feature maps before global average pooling and projection into the embedding vector
Input image	Preprocessed signature image used during inference	Ensures Grad-CAM reflects the same input representation used by the model
Heatmap normalisation	Min-max normalised to [0, 1]	Converts the raw activation values into a consistent display range
Overlay method	Resized and overlaid onto input image	Aligns low-resolution activation map with visible signature regions

Since Grad-CAM is a qualitative interpretability method, the generated heatmaps were not treated as definitive causal proof. Instead, they were used as diagnostic evidence to support discussion of stroke-focused attention, shortcut learning, and possible failure modes

CHAPTER 5 EXPERIMENT/SIMULATION SETUP

5.1 Hardware Setup

The primary hardware involved in this project was a desktop computer. This computer was used to design, implement, and execute the deep learning model.

Table 5.1.1 Specification of Computer

Description	Specifications
Model	Gigabyte Z790 UD AX
Processor	Intel Core i7-14700K
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 4070 SUPER 12 GB GDDR6X
Memory	32GB DDR5 RAM
Storage	512GB SSD

5.2 Software Setup

5.2.1 Development Environment and Primary Language

The main language used was Python, a widely adopted language in machine learning and deep learning R&D projects. Python was chosen for its extensive ecosystem of open-source libraries available through PyPi, which significantly accelerated experimentation and prototyping without requiring developers to implement low-level utilities from scratch. The specific version of Python utilised was 3.13.9. Table 5.2.1.1 details the specifications of the development environment and version control tool.

Table 5.2.1.1 Development Environment

Categories	Name	Description	
Text Editor	**Visual Studio Code**	**Version**	1.109.2 (user setup)
		Purpose	Development and testing
Version Control	**Git**	**Version**	2.53.0
		Purpose	Version control, and as public repository for source code
Interactive Notebook	**Jupyter Notebook (Visual Studio Code)**	**Version**	Bound to the version of Visual Studio Code
		Purpose	Development and testing of the model

5.2.2 Deep Learning Framework and Libraries

Table 5.2.2.1 details the key frameworks and libraries used to define the logic behind dataset preprocessing, model definition, training, evaluation and testing, and visualisation of training and evaluation results.

Table 5.2.2.1 Deep Learning Framework and Libraries

Categories	Name	Description	
Framework	**PyTorch**	**Version**	2.9.0+cu130 (Dependent on the available GPU)
		Purpose	Main framework of this project. Contains pre-trained models for transfer learning
Libraries	**NumPy**	**Version**	2.2.6
		Purpose	Preprocessing of fetched dataset
	Scikit-learn	**Version**	1.7.2
		Purpose	Division of dataset into training, validation and testing
	Matplotlib	**Version**	3.10.7
		Purpose	Create visualisations of performance metrices
	Seaborn	**Version**	0.13.2
		Purpose	Create visualisations of performance metrices
	TorchVision	**Version**	0.24.0+cu130 (Dependent on the available GPU)
		Purpose	Provides image transformation tools for computer vision purposes
	Pillow	**Version**	11.3.0
		Purpose	Image preprocessing
	GradCAM	**Version**	1.5.5
		Purpose	Explainable AI

5.2.3 Deployment Tools

For demonstration and inference, the models were deployed using Streamlit, a library commonly used to quickly turn Python scripts into shareable web applications without any front-end languages and tools. The library allowed quick integration of model details on the site. No database-related tools, libraries, or frameworks were utilised during the development of this demonstration.

Table 5.2.3.1 Deployment Tools

Categories	Name		
Library	**StreamLit**	**Version**	1.53.0
		Purpose	Deployment, demonstration and inference

5.3 Settings and Configurations

5.3.1 Distribution and Setup

The files for both deep learning model and the deployment of the model were publicised in GitHub.

Deep learning pipeline: https://github.com/HappyPotatoHead/signature-verification-sct-plus

Deployment: https://github.com/HappyPotatoHead/sct-signature

Methods to install the source code of this project:

1. **CLI**
   a. **Clone the repository**
      This method requires an existing personal token or an SSH connection to the device.

```
PS C:\Users\Jimmy Ding\Documents> git clone https://github.com/HappyPotatoHead/signature-verification-sct-plus
```

Figure 5.3.1.1 Cloning GitHub Repository

   b. **Change directory into the project folder**
2. **GitHub Web**
   a. **Download the source code directly**

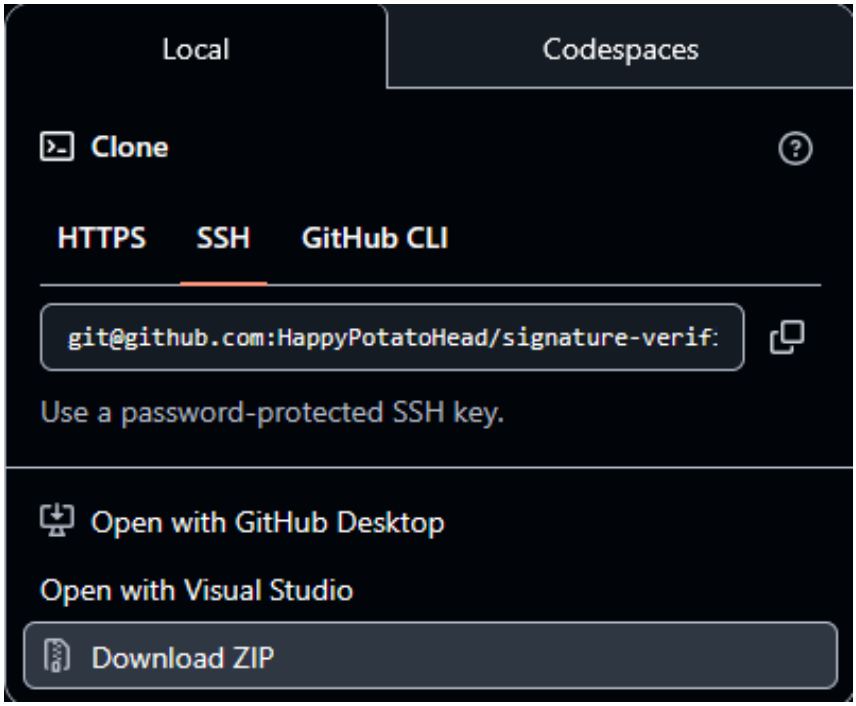


Figure 5.3.1.2 Direct Download GitHub on Website

b. **Unzip the download folder to the desired location**

Setting up development environment

1. **Create a local virtual environment**
   a. **Visual Studio Code**

      Visual Studio Code provides a simpler way to create virtual environments

      - Open the "Quick Open" dialog with Ctrl + P
      - Type ">" followed by "Create Environment"
      - Select the option under Python

   b. **CLI**

      - Execute the following command in the project folder: `python -m venv.venv`
      - Activate the virtual environment with `.\.venv\Scripts\activate.bat`

2. **Run** `pip install -r requirements.txt`

5.3.2 Runtime Configurations

The system configuration was externalised into a dedicated configuration script, which separated experimental parameters from the core implementation logic. This allowed the system to be executed under different conditions without modifying the source code. The configuration covered data preprocessing, training behaviour, optimisation strategy, and deployment settings.

Image preprocessing components, such as contrast enhancement, blurring, and thresholding, were parameterised to support modular and flexible experimentation. Specifically, Contrast Limited Adaptive Histogram Equalisation (CLAHE) was configurable via clip limit and tile grid size parameters, which improved image contrast while avoiding the over-amplification of noise.

During training, geometric augmentation, including rotation, translation, scaling, and cropping, was applied to simulate realistic variations in handwritten signatures caused by scanning conditions and writing posture. Additionally, only deterministic resizing and normalisation were applied during evaluation to ensure consistency. Training parameters could also be configured to match available hardware resources, as detailed in Table 5.3.2.1.

Table 5.3.2.1 Configurable Training Parameters

Parameter	Value	Purpose
Batch size	48	Balances GPU usage and training speed
Number of epochs	50	Provides sufficient convergent opportunities
Initial learning rate	0.001	Effective fine-tuning of pre-trained weights
Early stopping patience	10	Prevents overfitting
Execution device	CPU/GPU	Enables hardware-aware execution
Checkpoint directory	Configurable	Stores best-performing model's information
Log directory	Configurable	Records training progression for analysis

The optimiser and scheduler parameters were saved in dictionaries, where each key mapped directly to the parameter definitions within the PyTorch API (Table 5.3.2.2 and Table 5.3.2.3).

Table 5.3.2.2 Optimiser Parameters

Category	Parameters	Values
Optimiser	**Optimiser name**	AdamW
	Weight decay rate	0.001

Table 5.3.2.3: Schedulers Parameters

Scheduler Name	Parameters	Values
Sequential scheduler	**Milestone**	5
Linear scheduler	**Start factor**	0.1
	Total iterations	5
Cosine decay scheduler	**Maximum number of iterations**	Five epochs prior to the completion of the training cycle
	Minimum learning rate	0.000001

The system also supported configuring EfficientNetV2 variants as the model backbone, allowing trade-offs between computational cost and representational capacity.

A full sensitivity analysis was not conducted in this project due to computational and time constraints. This analysis is non-trivial because triplet-based DML depends on the sampling of informative positive and negative relationships. [18] note that a dataset with N samples can yield N^3 possible triplets, making the selection of effective samples a central training challenge.

Deep metric learning differs from standard classification because the objective is to learn an embedding space where distances reflect semantic similarity as noted in [19]. Consequently, loss design, sampling strategy, and pair/triplet construction directly influence the learned geometry. This ensured that the comparison between hard triplet mining, semi-hard triplet mining, and L_{SC+} was not confounded by different hyperparameter settings. Since the main objective was to isolate the contribution of the proposed L_{SC+}loss function, the same training configuration was kept fixed across all evaluated model variants. Table 5.3.2.4 shows the chosen hyperparameters and their rationale.

Table 5.3.2.4 Hyperparameter Choice Rationale

Parameter	Role	Value Used	Rationale	Expected Effect If Increased
Margin	Controls the separation boundary between genuine and forgery pairs	0.5	Kept fixed for fair comparison. 0.5 provides a middle ground	Stricter separation, but harder optimisation
μ	Weight positive pull regularisation	0.5	Prevents genuine signatures from drifting too far apart. 0.5 provides a middle ground	Tighter genuine clusters, possible over-compression
λ	Weight hard-negative penalty	1.0	This was used by the original paper	Stronger hard-negative separation, possible instability
P/K/F/M	Controls number of samples per update. Determines the batch size	$8 \times (2 + 2 + 2)$	Balanced GPU memory limits and pair diversity	More diverse training signal, higher computational cost

5.3.3 Deployment

To demonstrate the utility of the trained deep learning model, a web-based inference application was deployed using Streamlit, an open-source Python library designed for the rapid deployment of machine learning prototypes. To make the system accessible for external evaluation and demonstration, the application was deployed via the Streamlit Community Cloud, in addition to the publication of the source code.

5.3.4 License

Both the source code for the deep learning model and its deployment were published under the Apache 2.0 license, a permissive open-source license. This was chosen to provide legal clarity regarding the custom loss function developed. It ensured the novelty remained accessible for academic and commercial use while protecting both authors and users from future litigation.

Key Provisions

- **Commercial Use:** Individuals and companies were free to use, modify, and distribute the software for any purpose, including commercial applications.
- **Redistribution:** When modifications were made by a third party, they were required include a copy of the license and preserve all original copyright, patent, and trademark notices.
- **Notice of Changes:** If a user made significant changes to the source files, they were required include prominent notices stating that the files had been changed.

5.4 System Operation

5.4.1 Preprocessing

To ensure the model performed consistently across various datasets, each dataset used underwent similar preprocessing steps. Maintaining the established modularity, the preprocessing pipeline was divided into several modules. The pipeline began with the retrieval and sanity-checking scanned signature images.

This module supported formats such as Portable Network Graphics (PNG), Joint Photographic Experts Group (JPG/JPEG), Bitmap (BMP), and Tagged Image File Format (TIFF). Support for diverse file formats ensured the pipeline could handle images scanned from existing applications or various sources with minimal friction. It was assumed that the images were in a single-channel format; where this was not the case, a safety mechanism converted them to a greyscale format.

Once the images had been fetched and verified, image normalisation and enhancements were applied. These included contrast enhancement via CLAHE, followed by Gaussian blurring to reduce image noise, and concluded with Otsu thresholding for background removal. Other preprocessing steps were intentionally deferred at this stage to preserve modularity and allow downstream components to adapt preprocessing strategies as required.

When image preprocessing was complete, the images were saved in separate, dedicated folders that partitioned genuine signatures and forgeries. Each sample was associated with its corresponding class label, signer identity, and sample count. This dataset abstraction allowed the preprocessing pipeline to remain independent of the

training and inference components, enabling consistent reuse across multiple experimental configurations. Additionally, maintaining separate copies of signature images ensured data integrity and provided flexibility for experimentation during both the preprocessing steps and during model execution phases.

Before executing the training and evaluation pipeline, these pre-processed images were modified once more to ensure they aligned with the PyTorch API and the pretrained model's input requirements via augmentations. Because each image were originally of different sizes and resolutions, they were resized to a fixed spatial resolution ($384\,x\,384$). These images were then subjected to geometrical modification, such as translation, rotation, scaling, and shearing, followed by random cropping. Pixel density values were subsequently normalised to a standard range prior to tensor conversion. These operations enforced uniformity across samples originating from different signers and acquisition conditions while mimicking real-world characteristics.

5.4.2 Training and Evaluation

For clarity, the following implementation details were presented in the sequential order of the development environment, mirroring the execution flow of the project's Jupyter Notebook.

Initialisation Phase

This pipeline began with the initialisation phase. During this stage, configurations were loaded, including user-defined types, abstract data types, and parameters for the training cycle, optimiser, and scheduler. Backbone definitions were established to allow for flexible architecture experimentation. Once these configurations were loaded, the project was seeded to ensure reproducibility and consistency across runs. Various classes and augmentation strategies were also instantiated at this time.

Dataset Loading and Sampling Phase

This was followed by the dataset loading and sampling phase, which adhered strictly to the PyTorch API for creating `Dataloaders.` A writer-independent split was implemented, ensuring that images were partitioned by signer ID to prevent data leakage. Subsequently, the $PKFM$ sampling technique was applied to the training set to balance the inclusion of genuine signatures, skilled forgeries, and random forgeries.

Finally, the `Dataloaders` for all three datasets were instantiated, facilitating the transition to the training cycle.

Training Phase

With the datasets prepared, training could be commenced. This project followed modern standard practices for training deep learning models:

1. Forward pass to compute embeddings.
2. Loss computation using loss functions.
3. Backpropagation and optimiser updates.
4. Scheduler steps to adjust learning rates.

Within each epoch, the model was evaluated, and performance was measured via AUC scores. If the model failed to improve after a specified number of epochs, training was forcibly halted via the early stopping mechanism; otherwise, a checkpoint was saved with each improvement. The implementation also integrated TensorBoard for visualising loss curves, gradient updates, validation results, embedding projections, and learning rate fluctuations.

Testing Phase

Upon reaching the finalised model state, a comprehensive evaluation was performed on the held-out test set in a threshold-free manner. The model mapped each signature image into an embedding space; these embeddings were then organised into genuine-genuine pairs and genuine-skilled forgery pairs to simulate real-world verification queries. This evaluation is split into several analytical modules:

- Computing cosine similarity to quantify the distance between vector representations
- Generating classification metrics, including Accuracy, AUC, EER, FPR, and TPR
- Selecting the optimal threshold for decision making

The evaluation phase also incorporated visualisers, such as ROC curves and a confusion matrix, to provide a quick understanding of the model's classification efficacy and error margins.

Grad-CAM

Under Grad-CAM, the same code used to prepare the datasets for evaluation was reused; however, the procedure was simplified to compute only the cosine similarity scores. To make meaningful insights, the top 10 highest and lowest scores in both genuine-genuine and genuine-skilled forgery pairs were visualised to inspect the model's decision-making integrity.

5.4.3 Inference and Demonstration

The demonstration site served as a platform for users to test the model without the need to execute the deep learning pipeline themselves. It functioned as a transparent evaluation platform, enabling users to interact with the models in a secure and controlled environment.

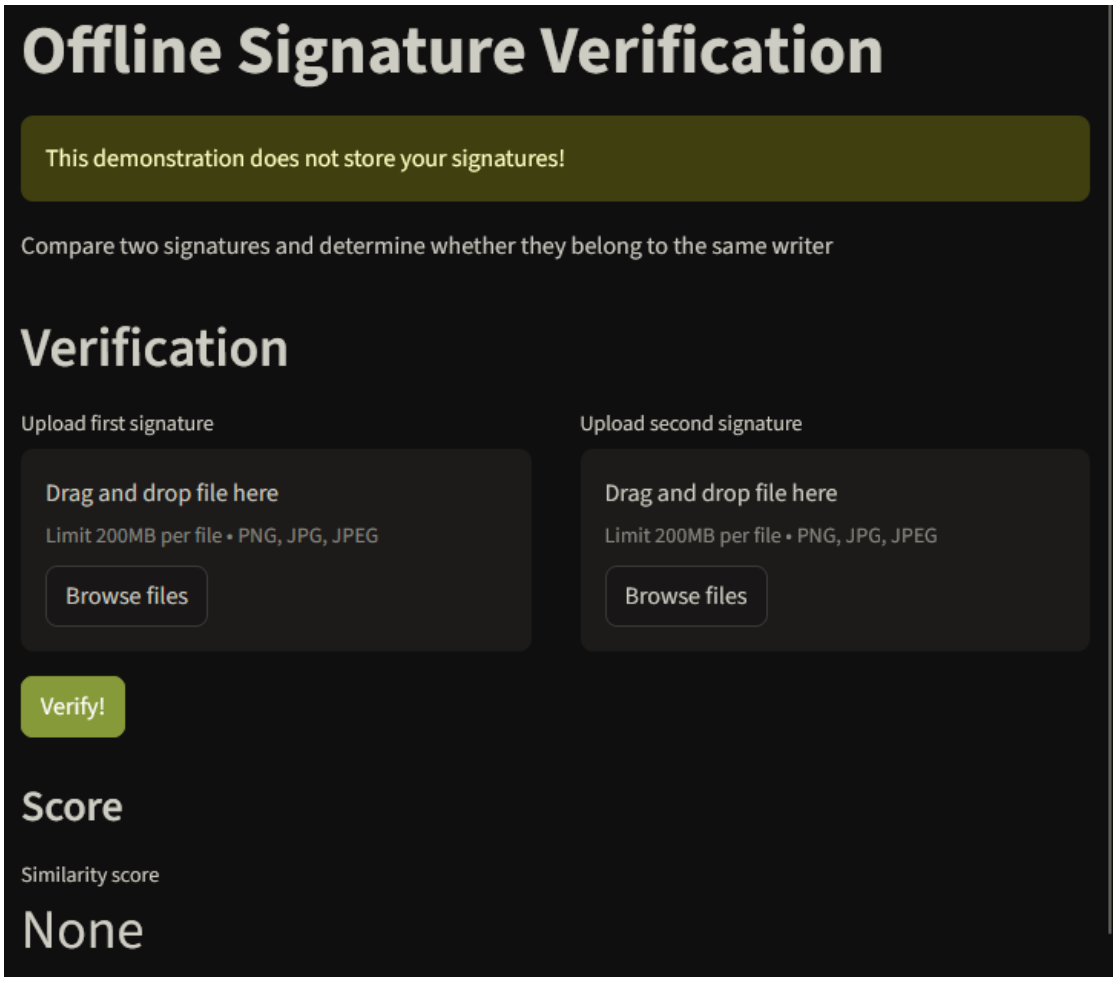


Figure 5.4.3.1 Interactive Deployment

To address data sensitivity, the demonstration platform operated on a stateless architecture, meaning no submitted information was persisted in any database or

storage system. Submitted signatures were processed in-memory for real-time inference. This design ensured users could test the system without risking biometric data leakage or privacy violations.

The site concurrently hosted three distinct model variants: those trained with hard triplet mining, semi-hard triplet mining, and the proposed L_{SC+}(Figure 5.4.2.2). This side-by-side availability provided immediate empirical evidence of the custom loss function's superior discriminative power.

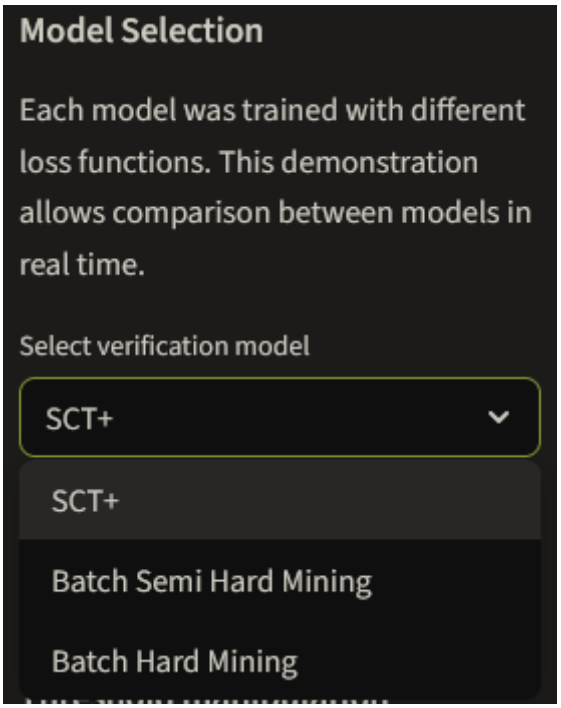


Figure 5.4.3.2 Real-Time Model Switching

A dynamic slider was also included to allow users to adjust the decision threshold in real-time. While each model defaulted to its calculated optimal threshold, this feature exposed users to the sensitivity of deep metric learning to thresholding. It demonstrated how shifting the boundary affected the trade-off between False Acceptance Rate (FAR) and False Rejection Rate (FRR).

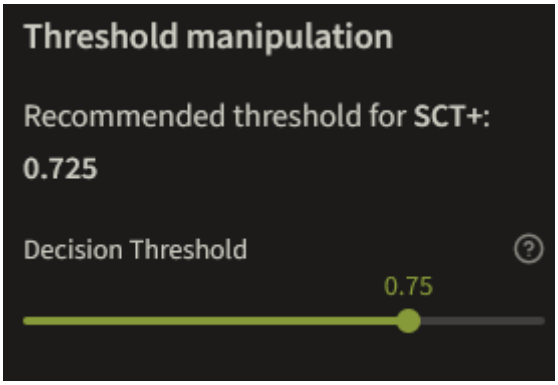


Figure 5.4.3.3 Threshold Slider

For users who were wary of providing their own signature, the site provided pre-loaded signature pairs. These pairs highlighted three possible scenarios: genuine-genuine pair (Figure 5.4.2.4), genuine-skilled forgery pair (Figure 5.4.2.5) and genuine-easy forgery pair (Figure 5.4.2.6).

Figure 5.4.3.4 Genuine-Genuine Signature Pair

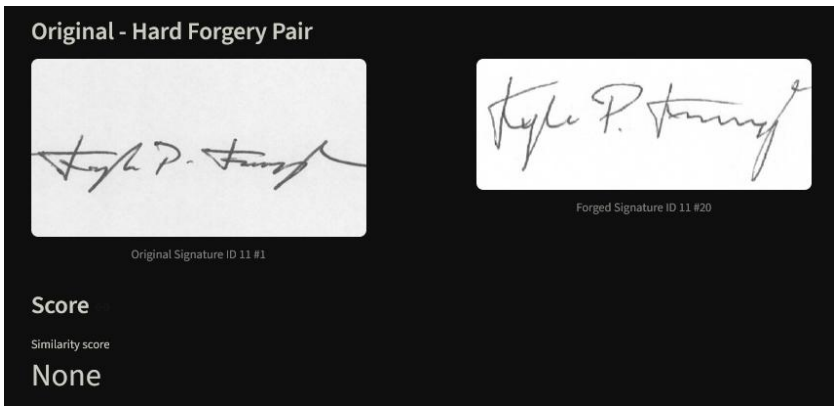


Figure 5.4.3.5 Genuine-Skilled Forgery Pair

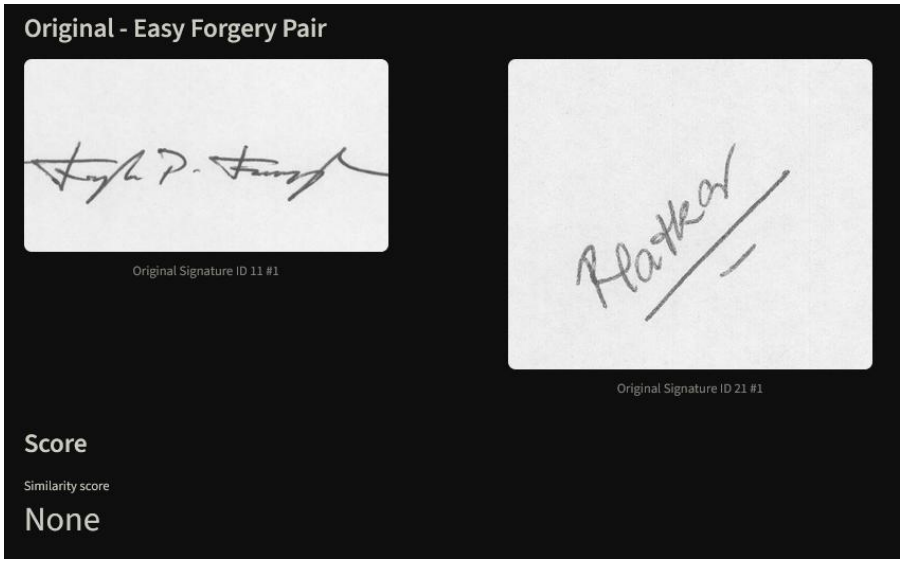


Figure 5.4.3.6 Genuine-Easy Forgery Pair

To ensure transparency regarding the model's efficacy and performance, the site provided technical justification on false classifications. Moreover, to further exemplify transparency, a "How it Works" section offered a high-level summary of the model's architecture, the L_{SC+} loss function, training protocols, evaluation, and design decisions, as well as design decisions.

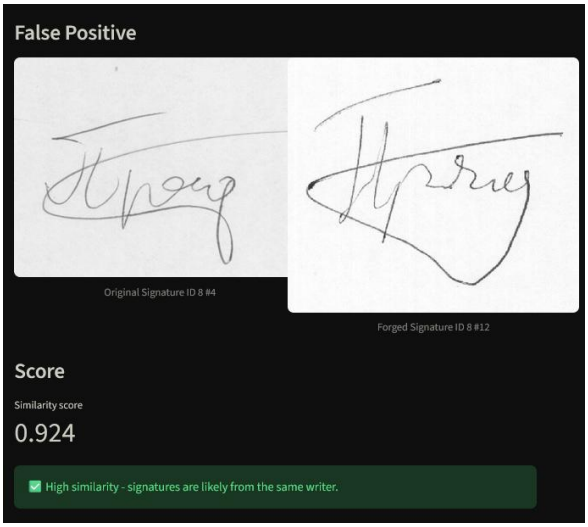


Figure 5.4.3.7 False Positive Example

Figure 5.4.3.8 False Negative Example

Ultimately, the demonstration site ensured that both academic research and practical utility were comprehensively covered.

Live Demonstration Link: https://sct-signature-demo.streamlit.app/

5.5 Implementation Issues and Challenges

Scarcity of Real-World Signature Datasets

As a form of biometric verification information, difficult, large, publicly available offline signature datasets were scarce, making them difficult to obtain in practice. Gathering thousands of samples to sufficiently train a robust model was unrealistic due to time constraints, privacy concerns from signers, and the severe security implications of leaking collected signatures. Consequently, this project relied on publicly available datasets, such as the CEDAR dataset. While these datasets provided sufficient signature images, they were limited in diversity, particularly in the number of samples per writer. Additionally, these datasets had been cleaned and curated for research purposes, which may have attenuated their real-world efficacy due to potentially inaccurate representation of daily signing conditions.

Hardware and Computational Limitations

Training deep learning models, even if transfer learning is utilised, required powerful GPUs with sufficient VRAM. While weaker GPUs could still be utilised, training was slower and resource constrained. Limited hardware restricted the batch size and slowed down experimentations, ultimately making training and extensive hyper-parameter tuning less efficient. To mitigate this, cloud-based solutions such as Google Colab, Kaggle, Amazon SageMaker, and Azure Machine Learning were considered, but they introduced constraints such as limited flexibility and control over the development environment, and limited free GPU availability. Additionally, the reliance on a stable Internet connection for cloud-based tools posed a potential risk to the project's progress.

Sensitivity of Metric-Learning Loss Functions

Loss functions such as L_{SC+} were highly sensitive to the margin parameter, learning rate schedulers, sampling strategies, and the quality of the embedding space at early training stages. Small changes in these hyperparameters could lead to outcomes ranging from minor variations to drastic failures, such as exploding gradients or the collapse of the embedding space. This stressed the need for careful, selective decision-making. Fortunately, by leveraging established research, various solid starting point for hyperparameters were identified.

Inter-Class and Intra-Class Variation Problems

Signatures exhibited high intra-writer variation and low inter-writer variation, complicating the creation of an embedding space that sufficiently separated different classes. This leads to:

- Difficulty producing meaningful positive pairs.
- Easy negatives dominate batches.
- Model overfitting to noises such as stylistic quirks.

To mitigate this issue, data augmentation was used to increase intra-class variation, creating variations; however, it was acknowledged that this approach could not fully replicate the natural complexity of handwritten signatures.

Generalisation Across Writing Styles and Scripts

As the model was trained primarily on Latin-alphabet signatures, its generalisation to logographic-based languages (such as Chinese, Japanese, and Korean) or Brahmic scripts was expected to be limited. Cross-script transfer remained an ongoing research challenge, as strokes, directionality, and density patterns differed significant across scripts, challenging the generalisability of the model.

Valid Triplet Availability During Training

Deep metric learning required a balanced distribution of hard positives and hard negatives in each batch. However, in real-world settings, offline signature datasets are often imbalanced, creating imbalanced batches. This can make training unstable, significantly slow convergence, and cause overfitting. Fortunately, this was managed by ensuring that the batch structure and sampling method were implemented carefully.

5.6 Concluding Remark

This chapter outlined the implementation of a development environment for training and testing a deep learning model. It reflected a well-thought-out structure, meticulously engineered to meet the established research questions and project goals. Logical hardware provisioning and software architecture demonstrated a strong technical framework capable of supporting complex pipeline requirements, reflecting the standards of real-world research and development.

Notable highlights included modular implementation of various functions in the pipeline, the flexibility to utilise different models and datasets, and the successful deployment of the model for demonstration.

While various challenges obstructed the project's progress, particularly regarding the sensitivity of the metric learning loss function, model generalisability, and potential imbalanced batches, these were addressed with logical solutions and designs. In conclusion, following best practices and research standards, this implementation served as a reliable baseline for future research and potentially for industry applications.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Verification and Validation Strategy

- **Accuracy Against Skilled Forgeries of The Same Person**
  - The model was be tested on its ability to successfully differentiate genuine signatures from forged ones within the same class. This verification was the most pertinent, as detecting skilled forgeries represented the defining challenge of signature verification.
- **Discrimination Between The Signatures of Different People**
  - The model had to strike a balance between clustering signatures of different individuals into distinct groups while maintaining the ability to differentiate forgeries within each group. This ensured that inter-class separation was preserved without compromising intra-class discrimination.
- **Robustness to Multiple Signing Styles**
  - The model was evaluated by its ability to handle natural variations in signature styles produced by the same individual. This was necessary to ensure that the system remained reliable even when users employed multiple signature styles.
- **Threshold Analysis**
  - Since verification decisions depended on a similarity threshold, the model was evaluated across a range of thresholds using ROC curves, AUC scores, and Equal Error Rate (EER). This ensured that performance was not biased by a single cutoff and provided a fairer comparison between models.
- **Scalability and Writer-Independence**
  - The model was assessed on its ability to enrol new signatures without re-training. This involved storing new reference signatures for each writer and verifying incoming signatures against them. Writer independence was necessary to support the goal of real-world deployment, where retraining for every user would have been unscalable in the long run.

6.2 System Performance Metrics

The performance of the proposed model was defined by its ability to generate a discriminant and robust embedding space for offline signature verification. The primary indicator of model success was its performance on unseen signatures. The model was expected to achieve at least 80% accuracy against skilled forgeries when evaluated at its optimal threshold on Latin-based signatures (e.g., Malay, Spanish, and German). This constraint was due to the training dataset, CEDAR [14][15], which contained exclusively English signatures; applying the model to logographic or Brahmic-based scripts was expected to produce significantly lower performance and was therefore not used for system benchmarking.

In addition to threshold-dependent accuracy, the system was evaluated using the Area Under the Curve (AUC). The model was expected to achieve an AUC of at least 90%, indicating strong separability between genuine and forged signatures. Unlike accuracy, AUC evaluated model behaviour across all possible thresholds, making it a more reliable metric for assessing the discriminative quality of the embedding space.

Beyond conventional verification metrics, the system performance was characterised by several qualitative and computational expectations:

- **Embedding Space Quality**
  - The model was intended to learn a compact, well-structured embedding space where genuine signatures from the same writer formed tight, interpretable clusters, while forgeries remained well separated.
- **Convergence Efficiency**
  - Training was expected to converge stably, without significant oscillations in loss curves or the collapse of embedding vector norms. This stability reflected the intended behaviour of L_{SC+}.
- **Robustness to Handwriting Variations**
  - The model was expected to handle common variations such as rotation, pen pressure, noise, or minor distortions. Preprocessing steps and

controlled image augmentations ensured that the model was trained for these scenarios.

- **Generalisation Capability**
  - While trained solely on Latin-script signatures, the system was expected to generalise consistently across languages within the same script family.

Together, these performance definitions established a clear set of quantitative and qualitative expectations that guided the design, training, and evaluation of the proposed signature verification system.

6.3 Testing Setup

6.3.1 Evaluation Signature Images Setup

Although the model was trained within a deep metric-learning framework using structured batch sampling, the testing phase followed a pair-based verification protocol. This distinction reflected the operational nature of real-world signature verification systems, where a queried signature is compared against a reference signature rather than evaluated in triplet form.

During testing, each signature image was first projected into the learnt embedding space before the similarity between them was calculated with cosine similarity. Subsequently, embeddings were organised into two primary pair categories:

- **Genuine-genuine pairs**: To establish baseline positive matches for each signer.
- **Genuine-skilled forgery pairs**: To simulate high-stakes negative verification attempts.

This pairing strategy simulated practical verification queries in which the system must determine whether a submitted signature matches its claimed identity. For each signer in the test set, genuine signatures were paired against other genuine samples belonging to the same signer. Conversely, skilled forgeries were paired against genuine signatures of the corresponding signer to construct negative verification attempts.

Genuine-Genuine Pair

Genuine Sample One

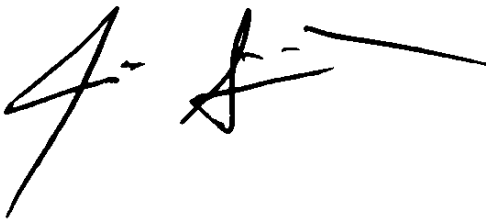


Figure 6.3.1.1 Genuine Sample One

Genuine Sample Two

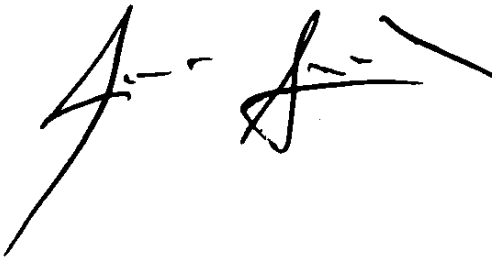


Figure 6.3.1.2 Genuine Sample Two

Genuine-Skilled Forgery Pair

Genuine Sample

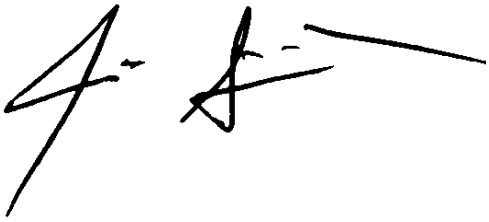


Figure 6.3.1.3 Genuine Sample One

Skilled Forgery Sample

Figure 6.3.1.4 Skilled Forgery Sample One

6.3.2 Threshold Selection

The proposed model produces a cosine similarity score for each pair of signature images rather than directly outputting a binary class label. A higher similarity score indicates that the two signatures are closer in the embedding space, while a lower similarity score indicates that the pair is more likely to belong to different classes. Therefore, a decision threshold is required to convert the similarity score into a verification score. If the similarity score is greater than or equal to the threshold, the pair is classified as genuine; otherwise, it is classified as a forgery.

The decision threshold was selected for evaluation purposes using the Equal Error Rate (EER) operating point. First, genuine-pair similarity scores and forgery-pair similarity scores were computed from the evaluation set. The Receiver Operating

Characteristic (ROC) curve was then generated by sweeping across possible threshold values. For each threshold, the False Positive Rate (FPR) and True Positive Rate (TPR) were computed. The False Negative Rate (FNR) was obtained as:

$$\mathrm{FNR} = 1 - \mathrm{TPR}$$

The selected threshold was the threshold where the absolute difference between FPR and FNR was minimised:|

$$t* = \arg\min_{t} |\mathrm{FPR}(t) - \mathrm{FNR}(t)|$$

At this operating point, the false acceptance and false rejection rates are approximately balanced. This threshold was then used to report the accuracy, true positive rate, false positive rate, and confusion matrix for each model.

It should be noted that this threshold is used as an evaluation operating point rather than universal deployment threshold. In a real-world deployment scenario, the threshold should be calibrated on a separate validation set according to application requirements. For example, a stricter threshold may be preferred when reducing false acceptance is more important, while a more lenient threshold may be used when reducing false rejection is prioritised. Since this project focuses on controlled model comparison, the EER operating point provides a balanced and consistent way to compare the different loss functions.

Table 6.3.2.1 shows the selected threshold for each evaluated model.

Table 6.3.2.1 Threshold Selection

Model		**Threshold Selection Method**	**Selected Threshold**
Backbone	**Loss Function**		
EfficientNetV2	Hard Triplet Mining	EER operating point	0.667
	Semi-Hard Triplet Mining		0.777
	L_{SC+}		0.725

The reported accuracy should therefore be interpreted as the accuracy obtained at the EER operating threshold. In contrast, the AUC score remains threshold-independent and is used to evaluate the model's overall ability to rank genuine pairs higher than forged pairs.

6.3.3 Grad-CAM

Using the same code used for the pipeline's evaluation phase, pairs of genuine-genuine and genuine-skilled-forgery signatures were generated. The evaluation function was simplified to return only the cosine similarity score of each pair along with their corresponding IDs. This list of results was then sorted by score in ascending order, enabling the introduction of four categories for analysis: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

Additionally, since analysing every pair would have been inefficient and would have resulted in diminishing return, only the top ten and bottom ten from the list of results were analysed. From the sorted list of genuine-genuine pairs, the last ten pairs provided the True Positives, while the first ten pairs represented the False Negatives. Conversely, from the sorted list of genuine-skilled forgery pairs, the last ten results indicated pairs False Positive flags, and the first ten pairs indicated True Negative flags.

6.4 Training Observations (CEDAR Dataset)

6.4.1 Hard Triplet Strategy

Using batch hard mining, the model demonstrated the ability to separate negative samples from positive ones. At the commencement of training, the mean of the Attraction Term (representing the distance between the anchor and genuine samples) was higher than the Repulsion Term (the distance to non-matching samples). By the end of the training process, the model successfully minimised the Attraction Term from approximately 1.4 to 0.7, while the Repulsion Term was consistently concentrated between 1.15 and 1.25. This trajectory confirms that the model effectively clustered genuine signatures together while displacing forgeries, achieving the desired embedding separation.

Attraction Term

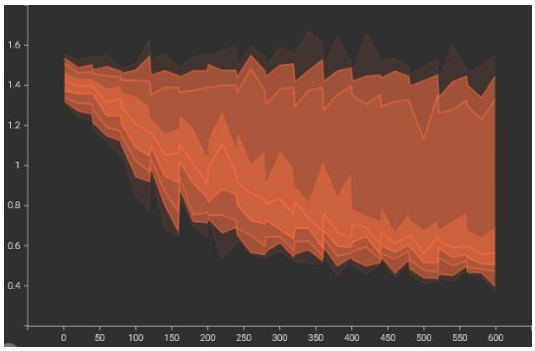


Figure 6.4.1.1 Hard Triplet Strategy Attraction Term Stacked Area Chart

Repulsion Term

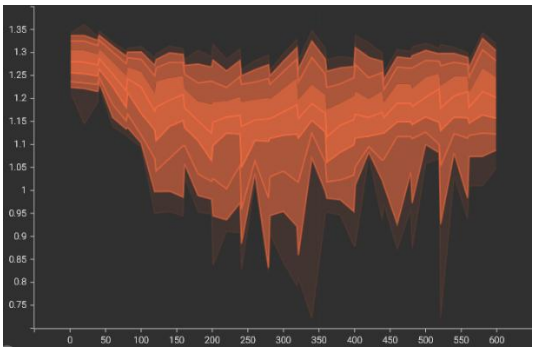


Figure 6.4.1.2 Hard Triplet Strategy Repulsion Term Stacked Area Chart

The histograms further illustrated these favourable dynamics: the distribution of the Repulsion Term remained concentrated on the right side of the axis, while the distribution of the Attraction Term shifted to the left. This increasing spatial separation between the two density peaks provides clear visual evidence that the model learned to distinguish between genuine signatures and forgeries, effectively widening the margin.

Attraction Term

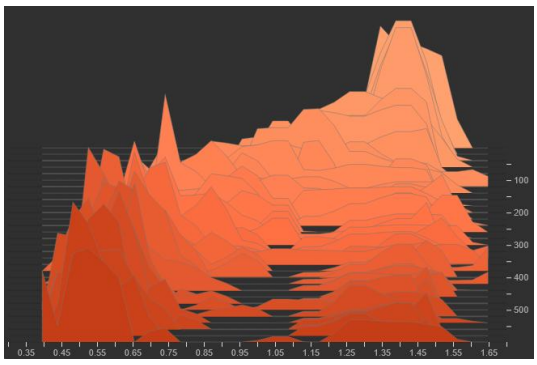


Figure 6.4.1.3 Hard Triplet Strategy Attraction Term Histograms

Repulsion Term

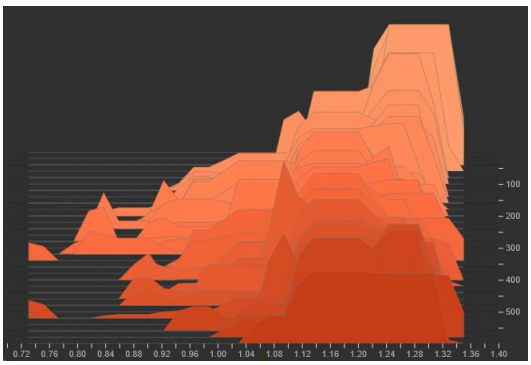


Figure 6.4.1.4 Hard Triplet Strategy Repulsion Term Histograms

When evaluated on the withheld test set, the model yielded a classification accuracy of 78.23% . This indicates that the system correctly identified signatures in approximately 78 out of every 100 instances, demonstrating acceptable performance for this specific evaluation context. The relatively high anchor-positive cosine similarity alongside a low anchor-negative cosine similarity confirms the model's capacity to discern forgeries from genuine signatures. Furthermore, an analysis across the entire threshold range identified 0.667 as the optimal threshold, at which point the misclassification rate was 21.77%.

Table 6.4.1.1 Hard Triplet Strategy Evaluation Results

Metrics (Threshold: 0.667)	**Results**
Accuracy	78.23%
True Positive Rate	78.20%
False Positive Rate	21.80%
AUC	0.8607
EER	21.77%

Similarity Score (Cosine Similarity)	**Results**
Positive Score	0.7784
Negative Score	0.3611

Hard mining produced smooth and globally consistent boundaries, as illustrated by the clean ROC curve in Figure 6.4.1.5. The curve yields an AUC of 0.8067, suggesting that the embedding space learnt under hard triplet mining maintains a strong degree of class separability across varying thresholds. However, the confusion matrix (Figure 6.4.1.6) reveals a more detailed picture; even at the optimal threshold, False Acceptance Rate remains relatively high (~21.8%). This indicates that a significant portion of skilled forgeries still resides within the acceptance margin of the genuine clusters.

These results demonstrate that while hard-negative mining successfully increases inter-class separation for the hardest violations, it does not guarantee robust generalisation across all variations of the same signature. Consequently, hard triplet mining alone may not suffice for real-world signature verification, necessitating further refinement or the integration of complementary strategies.

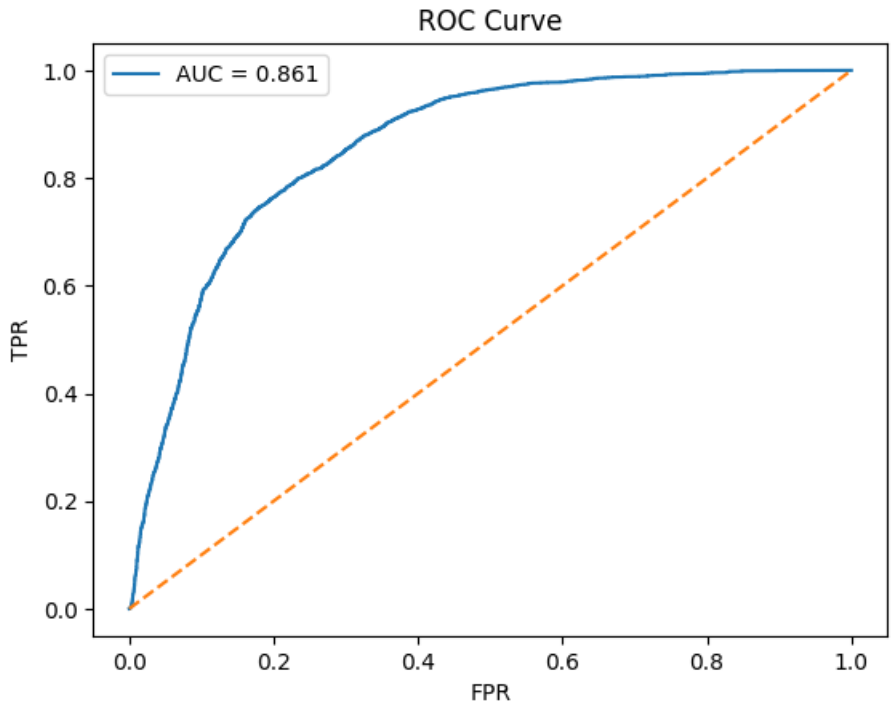


Figure 6.4.1.5 Hard Triplet Strategy ROC Curve

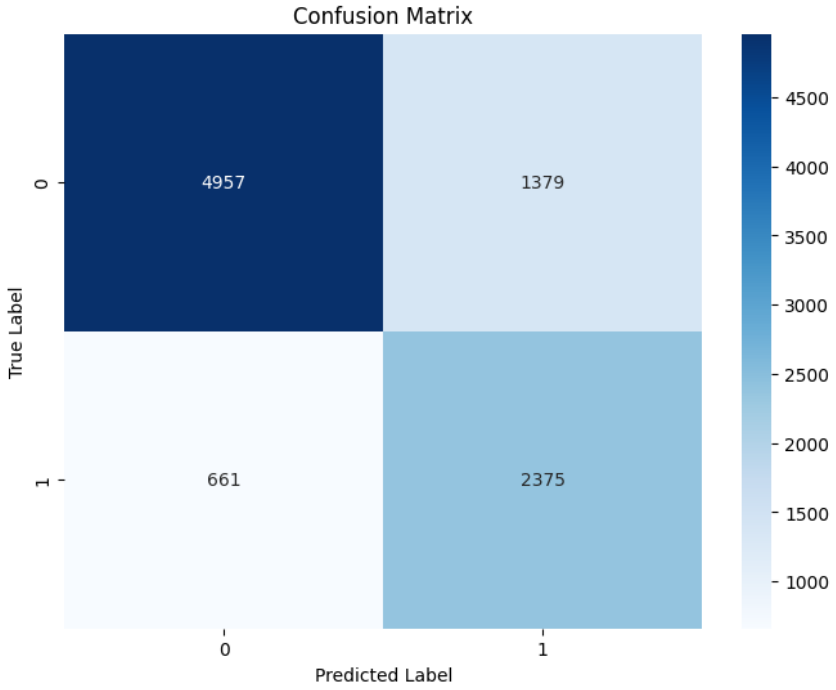


Figure 6.4.1.6 Hard Triplet Strategy Confusion Matrix

6.4.2 Semi Hard Triplet Strategy

Under the batch semi-hard mining strategy, the model demonstrated a capacity to separate negative samples from positive. At the start of training, the Attraction and Repulsion Terms exhibited similar means, indicating an initial lack of discriminative structure in the embedding space. By the end of training, the mean Attraction Term was successfully reduced from 1.1 to approximately 0.7, while the Repulsion Term stabilised between 1.1 and 1.2.

Attraction Term

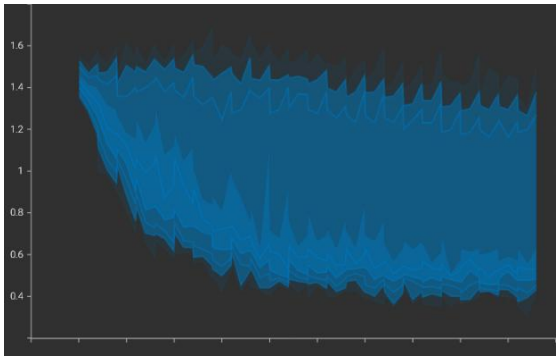


Figure 6.4.2.1 Semi Hard Triplet Strategy Attraction Term Stacked Area Chart

Repulsion Term

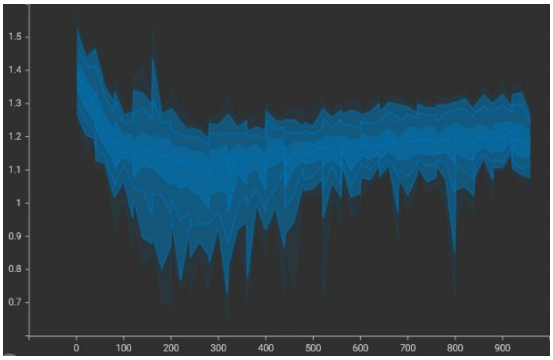


Figure 6.4.2.2 Semi Hard Triplet Strategy Repulsion Term Stacked Area Chart

Visualisations via histograms confirmed these desired changes. The distribution of the Attraction term shifted to the left, indicating tightening clusters. Although the Repulsion Term appeared to shift towards the centre rather than staying on the right, this is a visual noise caused by the disproportionately high distances recorded during the start of the training. Nonetheless, the Repulsion Term shifted to a stable range, ensuring a consistent, clearly defined margin.

Attraction Term

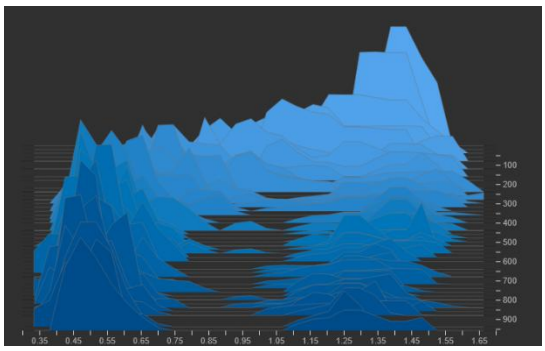


Figure 6.4.2.3 Semi Hard Triplet Strategy Attraction Term Histograms

Repulsion Term

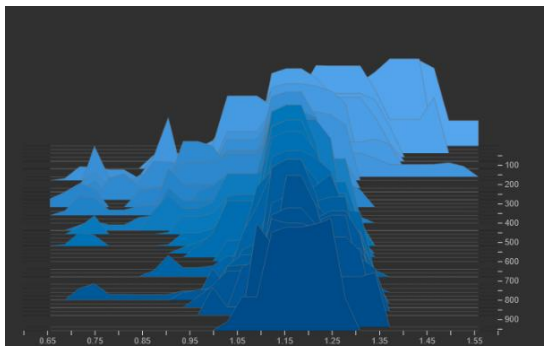


Figure 6.4.2.4 Semi Hard Triplet Strategy Repulsion Term Histograms

Upon evaluation with the test set, the model achieved a Recall rate of 82.10% and a False Positive Rate (FPR) of 17.90%. The high positive similarity scores and mid-range negative scores further validated the model's ability to cluster positives and their variants. Finally, at the optimal decision threshold of 0.777, the model yielded a Equal Error Rate of 17.88%.

Table 6.4.2.1 Semi Hard Triplet Strategy Evaluation Results

Metrics (Threshold: 0.777)	**Results**
Accuracy	82.12%
True Positive Rate	82.10%
False Positive Rate	17.90%
AUC	0.9071
EER	17.88%

Similarity Score (Cosine Similarity)	**Results**
Positive Score	0.8639
Negative Score	0.4967

Figure 6.4.2.5 shows a smooth and clean ROC curve with a high AUC score of 0.907. While the confusion matrix (Figure 6.4.2.6) still indicates some leakage, especially skilled forgeries that manage to bypass the threshold, the overall pattern suggests that semi-hard mining produces a balanced embedding space. This strategy did not create

a model that is overly sensitive to extreme outliers but rather one that focuses on samples that lie within the margin but not yet properly separated.

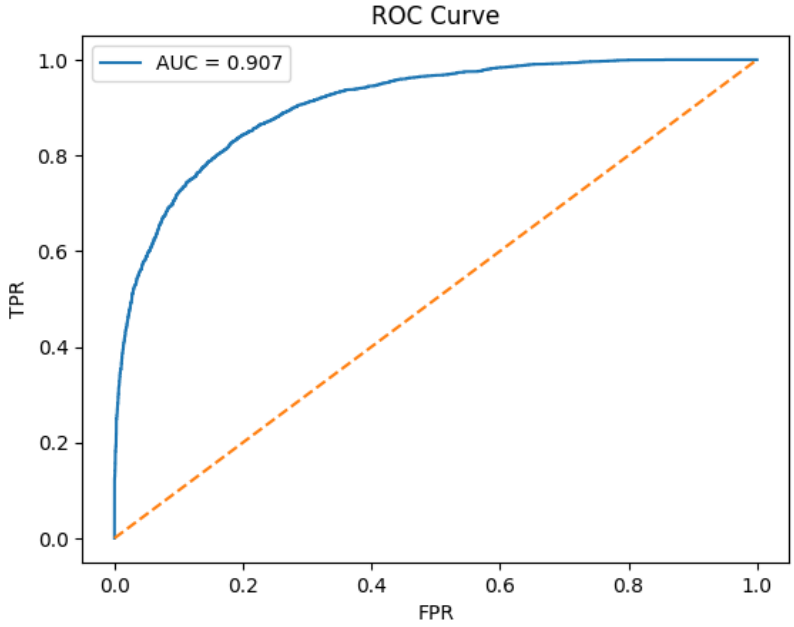


Figure 6.4.2.5 Semi Hard Triplet Strategy ROC Curve

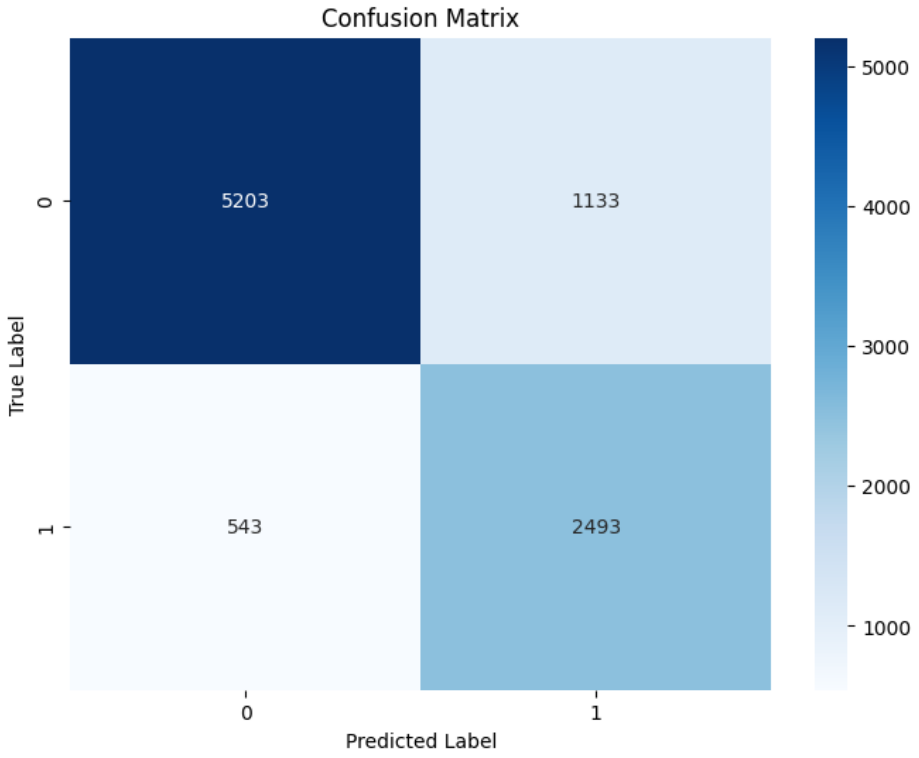


Figure 6.4.2.6 Semi Hard Triplet Strategy Confusion Matrix

6.4.3 L_{SC+} Loss Function

Unlike standard triplet loss, the L_{SC+} loss function utilises cosine similarity, effectively inverting the traditional distance-based intuition. In this architecture, the repulsion mechanism pushes hard negatives toward lower similarity values, while the explicit positive pull ensures that the attraction term increases, but only up to the specified margin. This prevents the cluster from collapsing into a single point while ensuring that positive clusters are not too sparse.

During training, the model exhibited the desired behaviour: a rising Attraction Term and a diminishing Repulsion Term. The histograms for the Repulsion Term were notably spread out, indicating that the model is complying with the intended behaviour of ignoring easy forgeries and aggressively pushing hard negatives.

Attraction Term

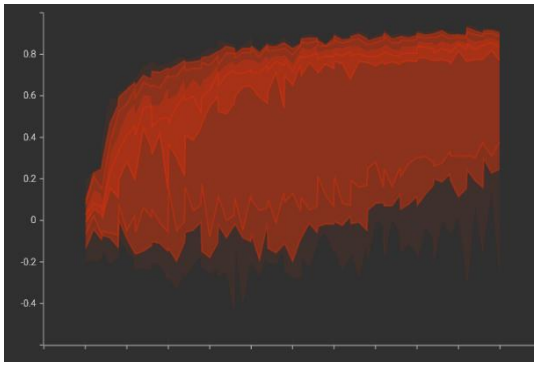


Figure 6.4.3.1 L_{SC+} Attraction Term Stacked Area Chart

Repulsion Term

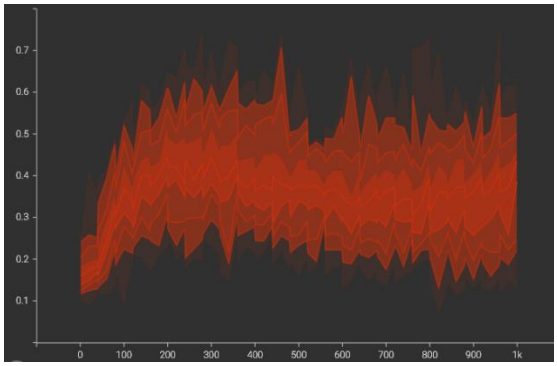


Figure 6.4.3.2 L_{SC+} Repulsion Term Stacked Area Chart

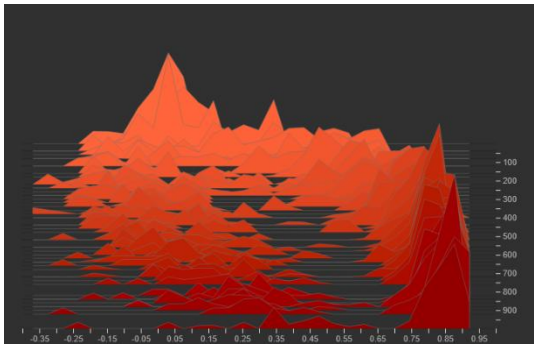


Figure 6.4.3.3 L_{SC+} Attraction Term Histograms

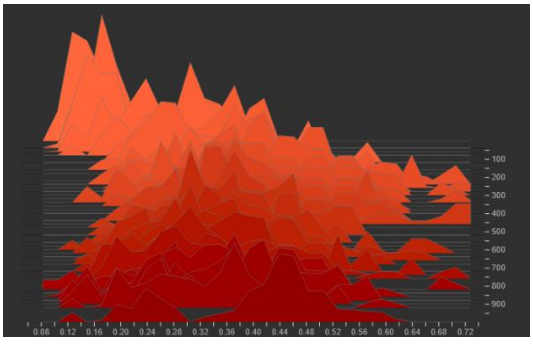


Figure 6.4.3.4 L_{SC+} Repulsion Term Histograms

On the test set, the model yielded the most promising results, achieving a Recall of 84.80%, a False Positive Rate of 15.20%, and an overall accuracy of 84.85%. At an optimal threshold of 0.725, a respectable Equal Error Rate (EER) of 15.15% was recorded.

Table 6.4.3.1 L_{SC+} Evaluation Results

Metrics (Threshold: 0.725)	**Results**
Accuracy	84.85%
True Positive Rate	84.80%
False Positive Rate	15.20%
AUC	0.9284
EER	15.15%

Similarity Score (Cosine Similarity)	**Results**
Positive Score	0.8444
Negative Score	0.3644

These metrics, alongside a superior AUC, point to a more discriminative embedding space than the triplet-based variants. The ROC curve (Figure 6.4.3.5) shows smooth global separation, while the confusion matrix (Figure 6.4.3.6) confirms consistent rejection of forgeries.

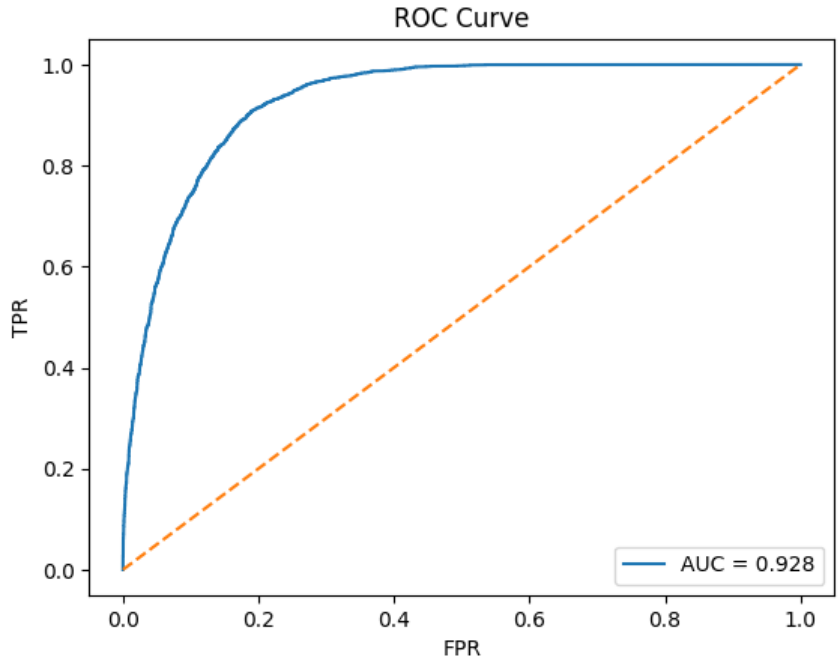


Figure 6.4.3.5 L_{SC+} ROC Curve

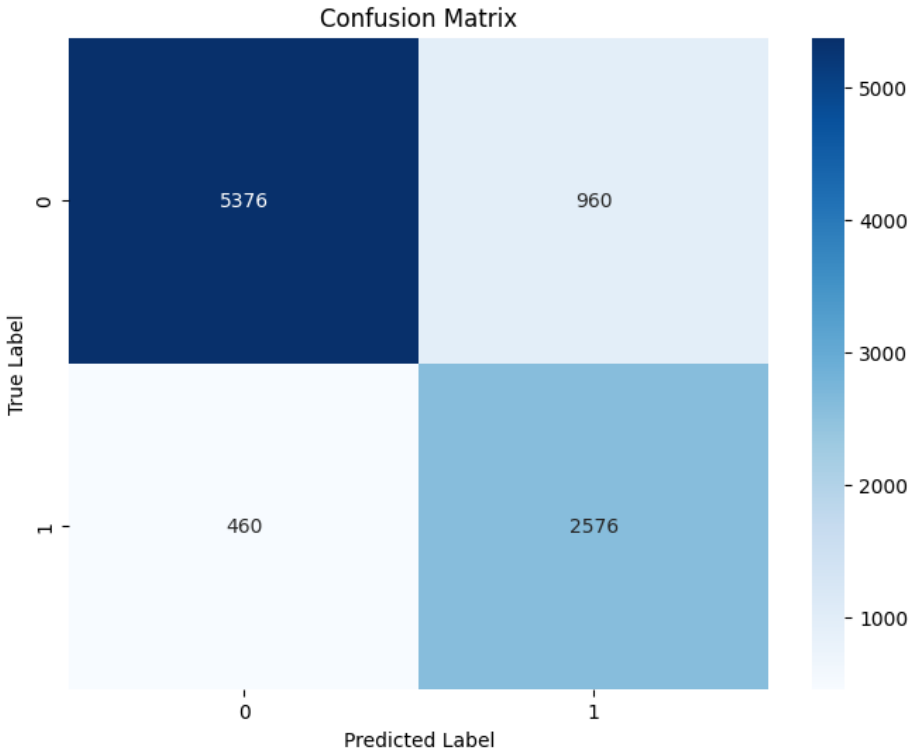


Figure 6.4.3.6 L_{SC+} Confusion Matrix

6.5 Performance and Objective Evaluation (CEDAR Dataset)

The performance of the proposed model was evaluated by its ability to generate a discriminative, robust embedding space for unseen signatures. The evaluation aimed to verify that the system achieves its primary objectives: high verification accuracy, generalisation across unseen writers, and effective separation of genuine and forged signatures.

6.5.1 Quantitative Results

Amongst the three loss functions, L_{SC+} performs the best. While some overlap persists, it was minimised compared to the other strategies. This is likely due to its trait of ignoring easy negatives, leading to closer, yet acceptable, distances. At the optimal threshold, this model yielded the lowest false acceptance of forgeries.

As shown in table 6.5.1, L_{SC+} scored the highest in TPR, AUC, and accuracy.

Table 6.5.1.1 Comparison of Results

Metrics/Loss Function (Threshold)	L_{SC+} **(0.725)**	$L_{\text{Semi Hard Triplets}}$ **(0.777)**	$L_{\text{Hard Triplet}}$**(0.667)**
TPR	84.8%	82.1%	78.20%
FPR	15.2%	17.9%	21.80 %
AUC	0.9284	0.9071	0.8607
Positive Score	0.8444	0.8639	0.7784
Negative Score	0.3644	0.4967	0.3611

Both positive and negative scores of L_{SC+} fall between those of semi hard (0.8639) and hard mining (0.7784). While initially counterintuitive, this highlights a critical point: a higher positive similarity does not necessarily translate to better verification. The margin between positive and negative distributions is what a model should prioritise. L_{SC+} achieves this by moderately enforcing intra-class compactness while consistently pushing hard negatives, creating more balanced clusters.

The ability to decouple anchor-positive from anchor-negative pairs allows L_{SC+} to receive independent gradient signals, ensuring continuous, informative updates regardless of triplet sampling quality. In contrast, the coupling effect in hard and

semi-hard triplet mining causes imbalance gradients, leading to overlap. The smoother gradient dynamics of L_{SC+} result in a more separable embedding space, improving discriminability across thresholds (higher AUC and lower EER) (Figure 6.4.3.5).

The operating threshold for L_{SC+} falls between semi-hard and hard mining, reflecting a more balanced embedding distribution. By comparison, the lower threshold required by hard triplet ming reflects its tendency to over-compress positives, which shifts the negative distribution closer and increases false acceptance of forgeries.

Overall, the results indicate that L_{SC+} produces more stable, generalizable, and discriminative embeddings. Despite not achieving the highest raw positive score, its improved separation of positive and negative distributions leads to superior AUC, TPR, FPR, and accuracy. This robustness arises from its decoupling of anchor-positive and anchor-negative contributions, which ensures continuous informative gradients and balanced embedding clusters.

6.5.2 Embedding Behaviour and Analysis

Across all loss functions, the models exhibited the desired behaviour: a leftward shift of anchor-positive distance (Figure 6.4.1.3, Figure 6.4.2.3, Figure 6.4.3.3) and a lingering anchor-negative distance on the right (Figure 6.4.1.4, Figure 6.4.2.4, Figure 6.4.3.4).

However, upon scrutiny, some nuances can be observed; there exist overlapping between the attraction and repulsion terms. The extent of this overlap and resulting stability varies significantly by strategy.

Hard triplet mining showed the most severe overlap between positive and negative distributions (Figure 6.4.1.3, Figure 6.4.1.4). This resulted in a higher rate of false predictions, even at the model's optimal performance threshold (Table 6.5.1, Figure 6.4.1.6). This aligned with the observation made by [10], who noted that an over-focus on hard triplets often causes model collapse. The excessive penalty on hard negatives forced the model to over-compress anchor-positive pairs to compensate. This over-fitting to specific hard samples reduced the generalisability of the model to unseen signatures from the same signer (Figure 6.4.1.6). The imbalanced focus on

hard negatives resulted in inconsistent separation of anchor-negative pairs, leaving some negatives closer to the anchor than desired.

Semi hard triplet mining exhibited less drastic overlap. This is likely due to its enforcement of separations based on $D_{ap} + \alpha$ rather than purely focusing on the hardest violations. However, this strategy can backfire when the threshold for forgeries falls within $D_{ap} + \alpha$ range, as seen in Figure 6.4.2.6, where false predictions remain relatively high. Notably, its reduced emphasis on hard triplets leads to slightly more dispersed anchor-positive pairs, increasing robustness to intra-class variation by allowing natural differences in genuine signatures (Figure 6.4.2.6).

6.6 Grad-CAM

6.6.1 Shortcut learning Effect

By isolating the top ten samples and applying Grad-CAM to interpret the model's output, diagnostic quirks within the model's logic were observed. These visualisations provided a transparent view of how the deep learning architecture interpreted features, revealing specific behavioural patterns in how it distinguished genuine from forged signatures.

According to [20], many of the failures or successes of deep learning models can be traced to one major factor, the preference for shortcuts over the intended solution. These shortcuts are often spurious and easy-to-exploit correlations. Such effect was found in certain pairs, as evidenced in Figures 6.6.1.1 and 6.6.1.2. The individual heatmaps showed the model bypassing stroke characteristics in favour of spurious background features. One plausible reason for this phenomenon was that the loss function forced discrimination in the embedding space, even when the model failed to discern meaningful biometric differences between certain pairs. As a result, the model resorted to minor background artefacts as a proxy to influence distances, even if these artefacts had been largely mitigated.

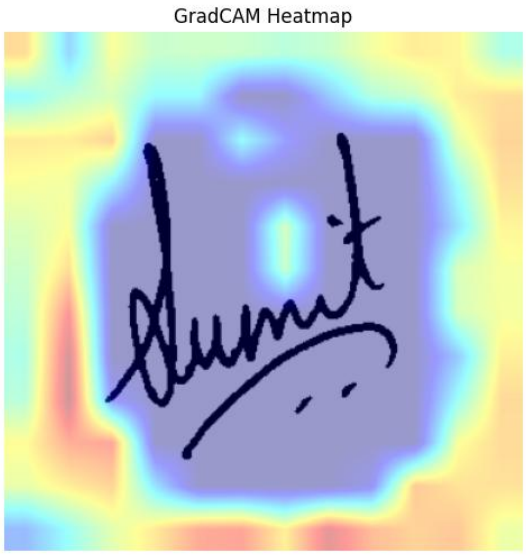


Figure 6.6.1.1 Signer 19 Shortcut Learning

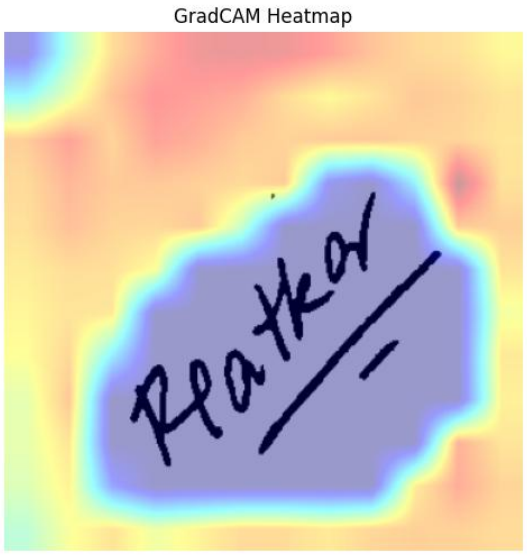


Figure 6.6.1.2 Signer 21 Shortcut Learning

While the complete elimination of shortcut learning was often unfeasible, binarisation served as a critical defensive measure to standardise the image background. By minimising paper-based artefacts, this process provided a sanitised input that encouraged the model to extract biometric features solely from signature strokes, thereby significantly reducing the influence of spurious scan-level data.

Despite these observations, it was noted that this shortcut learning behaviour was not a universal characteristic of the model's decision-making process; among the top 10 highest-confidence pairs, the effect was observed in only three cases. This suggested that the model's high ROC-AUC performance was primarily influenced by biometric stroke characteristics, although occasional reliance on scan-level artefacts occurred. This distinction was vital; it demonstrated that, while the model occasionally fell back on scan-level shortcuts in specific edge cases, its primary mode of verification remained focused on the structural characteristics of the handwriting.

To illustrate that the model was also capable of feature-based decision making, the following figures, Figure 6.6.1.3 and 6.6.1.4, illustrated instances of legitimate classification for Signers 9 and 52.

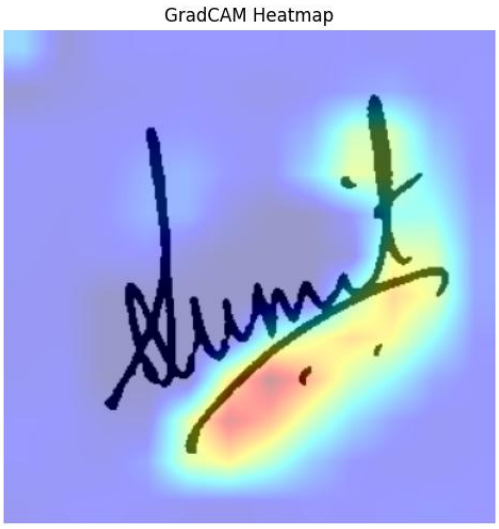


Figure 6.6.1.3 Signer 19 Genuine Signature

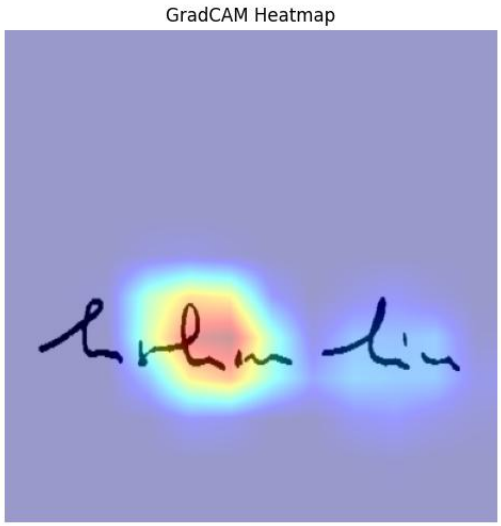


Figure 6.6.1.4 Signer 52 Genuine Signature

Furthermore, the Grad-CAM visualisations demonstrated that the data augmentation pipeline, particularly the geometric transformations, was effective in exposing the model to various signature scales and orientations. The model's ability to correctly detect the strokes of the signature, despite significant height and spatial variation, suggested that the augmentation strategy successfully prevented overfitting to a fixed signature location. Figures 6.6.1.5 and 6.6.1.6 illustrate one such example. Additionally, it was observed that in Figure 6.6.1.5 the model ignored the dots in the image and focuses entirely on the signature.

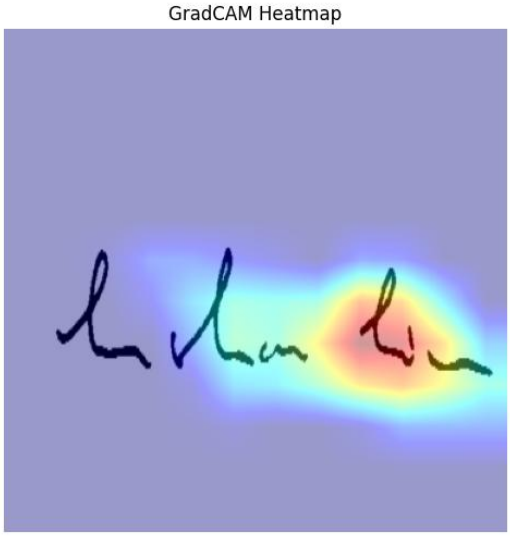


Figure 6.6.1.5 Signer 52 Genuine Signature

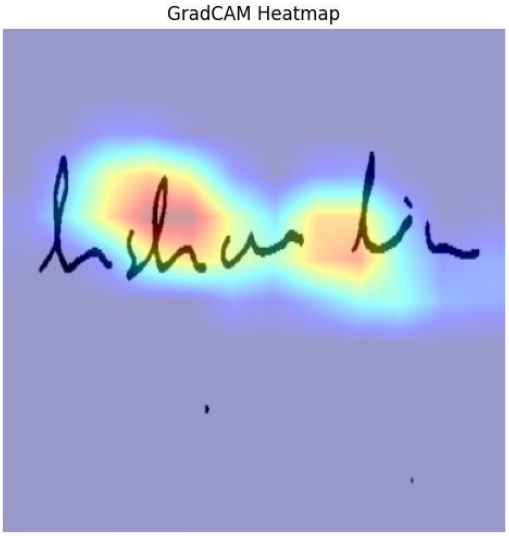


Figure 6.6.1.6 Signer 52 Forged Signature

6.6.2 Classification Visualisations

To further interpret the verification errors, Grad-CAM visualisations were reviewed together with the corresponding similarity scores and predicted labels. The observed cases were grouped into several qualitative categories. These categories do not represent exhaustive causal explanations; instead, they summarise recurring visual patterns observed from the heatmaps and error cases. Additionally, since Grad-CAM is qualitative and spatially course, these observations should not be interpreted as definitive proof of causation. Further ablation studies, such as removing small connected components, randomising signature placement, or cropping tightly around foreground strokes would be required to confirm the exact source of the shortcut behaviour.

True Positive and True Negative: Stroke-focused Correct Cases

In cases of true positives, the observation that can be made was that the heatmaps overlap in major stroke regions, as shown in Figure 6.2.2.1 and Figure 6.2.2.2. From the visualisation, the plausible explanation was that the model attended to meaningful biometric stroke patterns. This was also the case for false negatives, as shown in Figure 6.2.2.3 and Figure 6.2.2.4

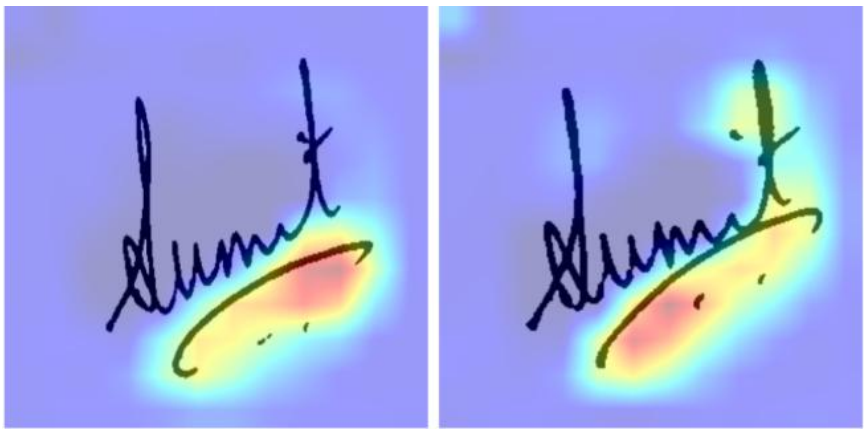


Figure 6.6.2.1 True Positive Grad-CAM Example One

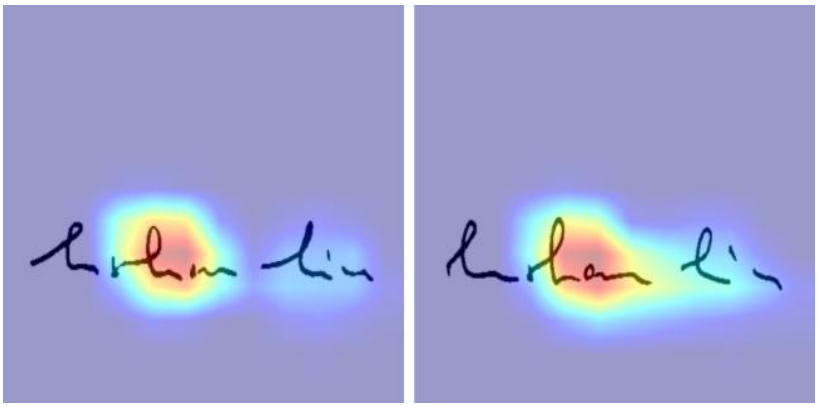


Figure 6.6.2.2 True Positive Grad-CAM Example Two

Figure 6.6.2.3 True Negative Grad-CAM Example One

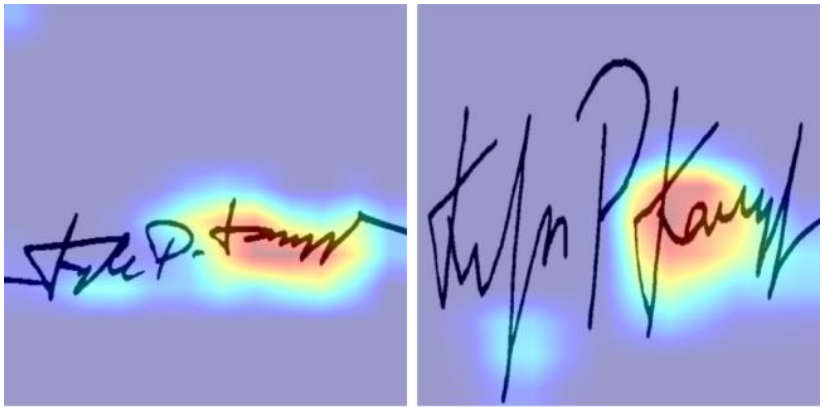


Figure 6.6.2.4 True Negative Grad-CAM Example Two

False Negative: High Intra-Class Variation

In such cases of false negatives, the natural variation in writing style, slant, spacing, or stroke shape causes lower similarity. The model seemed to have focused on different regions of the stroke, possibly due to the variation between the two signatures. Consequently, the heatmaps generated were diffused or focused on incomplete or noisy stroke regions and genuine signatures were rejected despite possessing the same signer id (Figure 6.2.2.5 and Figure 6.2.2.6).

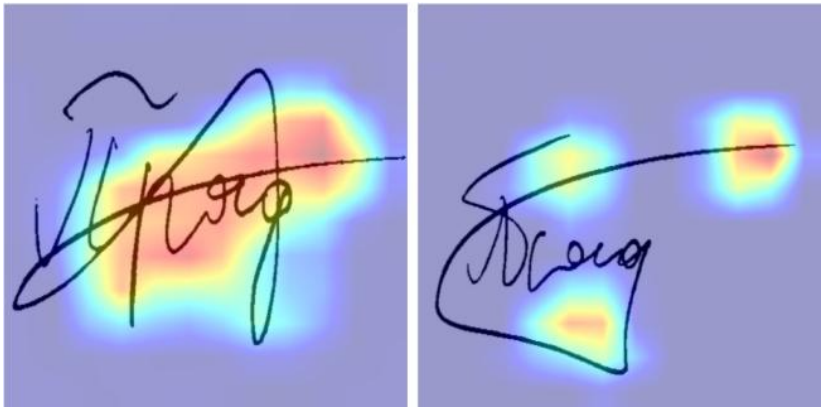


Figure 6.6.2.5 False Negative Grad-CAM Example One

Figure 6.6.2.6 False Negative Grad-CAM Example Two

False Positive: Artefact shortcut or Similar Stroke Pattern

The heatmaps shown focus on isolated non-stroke components when extracting feature representation. This may be explained with the model's reliance on local shortcut cues that correlate with similarity but are not robust biometric features.

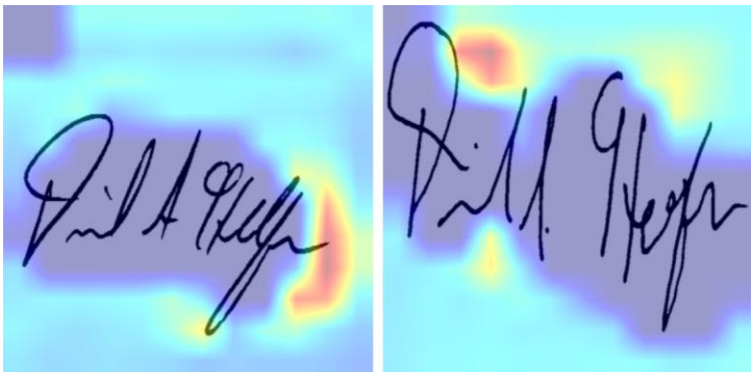


Figure 6.6.2.7 False Positive Grad-CAM Example One

Additionally, there were instances where the forgeries were accepted because the forged strokes visually resemble the genuine reference, close enough that it was accepted as a variation of the genuine.

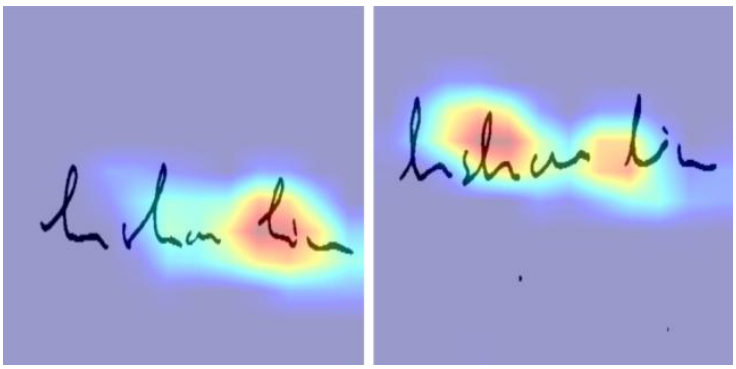


Figure 6.6.2.8 False Positive Grad-CAM Example Two

6.7 Evaluation on Self-Sampled Dataset

6.7.1 Quantitative Results

The following evaluation results were computed based on the dataset in Appendix B.

To determine the model's robustness to real-world scenarios, the model was evaluated on self-sampled signature dataset. As mentioned in Chapter 4.1.2, each participant was allowed to practice their forgeries before the final recording, creating a dataset that was potentially more difficult than the CEDAR dataset.

To ensure the best results, the threshold was recalibrated. Upon evaluation, the model achieved a slightly lower True Positive Rate (TPR), 83.3%, and conversely, a slightly

higher Equal Error Rate (EER), 16.46%. The overall negative score had down to 0.1646, highlighting the increase in difficulty of the dataset (Table 6.7.1.1).

Moreover, the high overall positive score (0.9450) and the low overall negative score indicated that the model was highly confident in clustering positive pairs; however, this came at the expense of pushing the threshold higher. This was not necessarily an issue, but it provided insight into how the model performed in a cross-dataset setting. Although the overall negative remained relatively low, the fact that the threshold had to be pushed so high suggested that some forgeries were also achieving high similarity scores. Consequently, to avoid a high rate of false positives, a stricter threshold had to be employed.

Table 6.7.1.1 Evaluation Results (Self-Sampled)

Metrics (Threshold: 0.887)	**Results**
Accuracy	83.63%
True Positive Rate	83.30%
False Positive Rate	16.20%
AUC	0.9450
EER	16.46%

Similarity Score (Cosine Similarity)	**Results**
Positive Score	0.9450
Negative Score	0.1646

Figure 6.7.1.1 denotes the model's discriminative capability in a confusion matrix. The model was largely successful in predicting True Negatives, with 67 correct predictions; however, nearly 35% (13/38) of forgery-genuine pairs were falsely accepted. To provide us with insights to these False Positives, Grad-CAM was implemented.

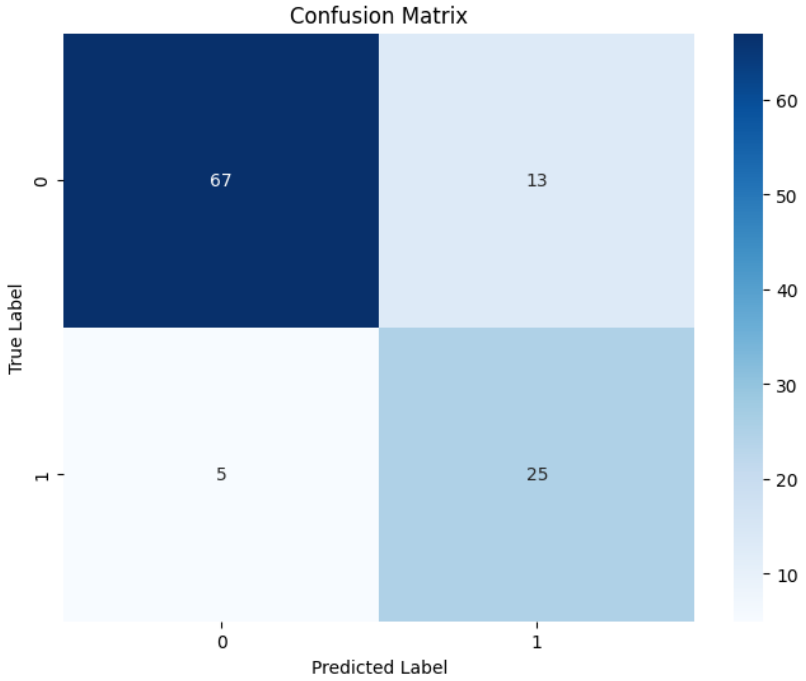


Figure 6.7.1.1 Confusion Matrix

6.7.2 Grad-CAM

The low count of false positives allowed us to plot all 13 pairs. The remaining pairs not shown in this chapter can be found in Appendix B.

In all of pairs involving signer 5, a common pattern was identified, the model appeared to have focused completely on the three specific dots on the top-right corner, ignoring other forgery indicators. Because these dots were an integral part of the signature, this phenomenon could not be flagged as a “shortcut learning” effect in the traditional sense, but rather as an optimisation of discriminative efficiency.

From the model’s standpoint, if a specific artifact (such as the three dots) was consistently present in each signature, it identified that artifact as a highly reliable feature. This caused a reduction in the model’s emphasis on more complex signature details. While efficient, this behaviour backfired in certain instances, most notably resulting in false positives. This may be attributed to the small volume of signatures per signer present in the training dataset; this constraint forced the model to rely on extreme, distinct markers rather than learning the full morphology of the signatures.

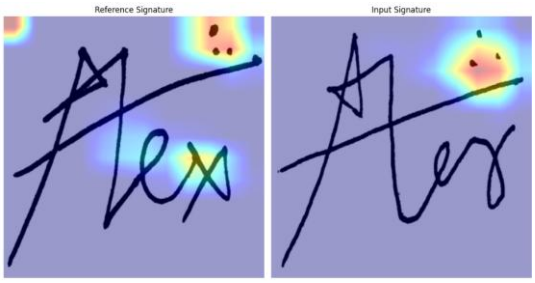


Figure 6.7.2.1 Signer 5 Grad-CAM Pair 1

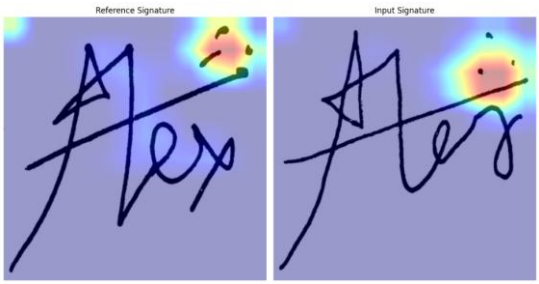


Figure 6.7.2.2 Signer 5 Grad-CAM Pair 2

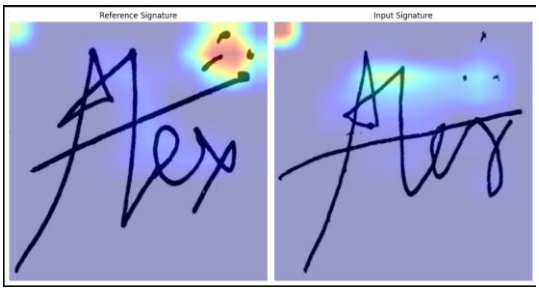


Figure 6.7.2.3 Signer 5 Grad-CAM Pair 3

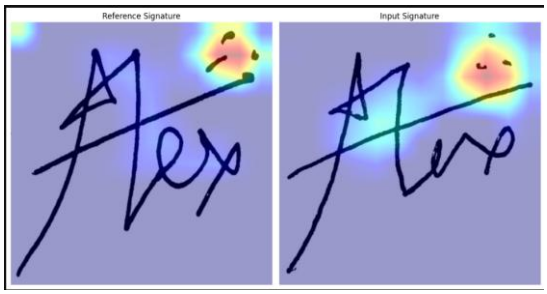


Figure 6.7.2.4 Signer 5 Grad-CAM Pair 4

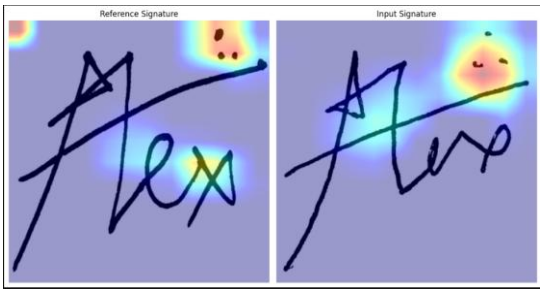


Figure 6.7.2.5 Signer 5 Grad-CAM Pair 5

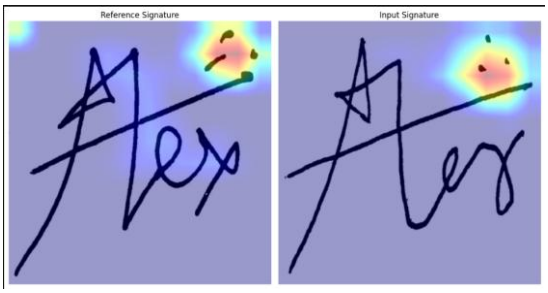


Figure 6.7.2.6 Signer 5 Grad-CAM Pair 6

Aside from the model's focus on the three dots, one other noteworthy observation was the model's fixation on the top left corner of the images. This phenomenon could be attributed to the tendency of Convolutional Neural Networks (CNNs) to encode implicit positional information, despite their theoretical translation-invariant design, as demonstrated in [21]. Such positional bias can arise from architectural factors, such as zero-padding, which introduced consistent boundary patterns that the model exploited during the learning process.

6.8 Project Challenges

Despite achieving encouraging verification performance, several significant challenges were encountered throughout the project. These challenges extended beyond implementation-level issues and involved experimental design, optimisation stability, research constraints, and practical deployment considerations.

A core challenge was designing a reliable and unbiased evaluation framework. Careful and thorough scrutiny of the implementation had to be carried out to ensure that no data leakage occurred from the training set to the test set. Offline signature verification differed from standard classification tasks in that performance had to be assessed based on writer-disjoint data splits and genuine-forgery pair comparisons.

Determining the appropriate depth and scope of experimentation posed a strategic challenge. Metric-learning frameworks involved multiple design choices, including backbone architecture, projection head dimensions, loss function selection, and data augmentation strategies. Conducting exhaustive experiments for every configuration was infeasible due to time constraints and dataset limitations. As a result, careful prioritisation was necessary to focus on experiments that maximised insight while maintaining project feasibility.

Although deep metric learning frameworks are well established in literature, implementing them in practice revealed several conceptual and stability-related challenges. Without careful analytical, mathematical, and thorough inspection of the code, the model was susceptible to embedding collapse, exploding or vanishing gradients, and slow learning. Bridging theoretical formulations with stable practical implementation required iterative experimentation and informed parameter selection guided by prior research. Detailed runtime profiling, including per-epoch training time, GPU memory usage, active pair count, and inference latency, was not explicitly logged during training. These metrics do not affect the reported AUC, EER, or accuracy, but they limit the extent to which computational efficiency can be quantified empirically. Future work should include systematic runtime and memory profiling.

While the system worked on a controlled dataset, deploying it in practical scenarios such as banks and offices would involve large-scale inference with many users. Managing inference time, memory, and ensuring robustness under concurrent usage becomes a non-trivial operational challenge. Additionally, such datasets are curated and cleaned for research purposes, which may not reflect real-world scenarios where signatures are messy, incomplete, or captured under varying conditions. Adapting the

model to handle real-world variability without overfitting to the curated datasets remained an ongoing challenge.

6.9 Concluding Remarks

In conclusion, the system evaluation confirmed that the model trained with the proposed custom loss function, L_{SC+}, and $PKFM$ sampling technique successfully achieved the proposed objectives, demonstrating strong performance across multiple metrices. By achieving higher AUC than models trained with the conventional triplet loss function and balanced intra- and inter-class separability, the system demonstrated a reliable ability to distinguish genuine signatures from forgeries, even for unseen writers. The model's performance was also validated on self-sampled signature datasets, which tested the model's robustness in real-world settings.

The results suggested that:

- L_{SC+} provided a stable and generalisable embedding space, reducing the risk of model collapse or overfitting.
- The $PKFM$ sampling technique was effective in ensuring fair and consistent batches across epochs, particularly when datasets were limited.

Furthermore, Grad-CAM analysis provided qualitative insight into the model's decision-making process. The primary finding was that the model resorted to paper artefacts and padding biases rather than stroke characteristics in certain cases; however, this was not universal, the model still demonstrated legitimate stroke-based feature learning, validating the effectiveness of the binarisation and data augmentation pipeline in mitigating shortcut learning. Additionally, it also revealed that the model resorted to the simplest and quickest features found in signatures when identifying characteristics. Overall, these findings validated the model's effectiveness, provided guidance for real-world integration, and highlighted remaining challenges for future improvements, such as cross-script generalisation and further optimisation of embedding compactness.

CHAPTER 7 Conclusion and Recommendation

7.1 Conclusion

This project developed a signature verification model that addressed key limitations in conventional offline signature verification systems. These limitations revolved around the struggle with intra-class variability and discerning genuine signatures from skilled forgeries. While most implementations relied on stacking multiple layers of models to achieve higher accuracy, resulting in larger, heavier, and possibly slower architectures. This limitation is compounded by the need for efficient computation in real-world use cases, as these models are frequently deployed on lower-powered machines.

The primary motivation for this project was to improve existing verification systems by providing a fast yet accurate model that could be refined without the need for additional complexity. By introducing the L_{SC+} loss function and integrating it with the state-of-the-art EfficientNetV2 backbone, the model achieved robust performance in distinguishing genuine signatures from forgeries, as demonstrated by consistently superior evaluation metrics.

In addition, the project extended PK sampling to $PKFM$ sampling strategy applicable to verification problems involving both positive and negative samples. The decoupling approach of L_{SC+} provided a principled and intuitive mechanism for separating anchor-positive and anchor-negative samples, reducing training variance and improving generalisation compared to traditional mining strategies. Even on limited data, L_{SC+} consistently outperformed semi-hard and hard triplet mining, proving its effectiveness in offline signature verification.

The model achieved strong results; testing on the held-out subset of the CEDAR dataset yielded an accuracy of 84.85%, a recall score of 84.80%, and an AUC of 0.9284. These results showed that the project's objectives were successfully met, especially in improving the performance of a deep learning model in this domain. Additionally, the mathematically lighter, more efficient formulation of the loss function demonstrated the model's potential for real-world system integration, particularly in high-stakes environments such as banks.

Additionally, the introduction of Grad-CAM provided insight into the model's decision-making process by visualising heatmaps. These heatmaps revealed the presence of shortcut learning, showing that binarisation did not eliminate it in controlled datasets, as the model occasionally resorted to scan-level artefacts as a proxy for biometric features. Grad-CAM also exposed the model's extreme efficiency, which is resorting to the simplest and defining characteristics of a signature

Despite the project's successes, limitations remained worth addressing, such as cross-script generalisation and the use of larger, real-world, messy signature images rather than cleaned, research-prioritised datasets. These limitations presented opportunities for researchers to further improve the integration of deep learning models in this domain. Overall, this work demonstrated strong potential for real-world applications in biometrics verification. The proposed contributions not only advanced signature verification research but also offered a scalable framework that can be extended to other biometric modalities, making it a valuable addition to the broader field of biometric security.

7.2 Recommendation

While the current model met the defined objectives, several notable areas remained that warrant further research to solidify its performance, efficiency, and effectiveness.

One key direction is cross-script or cross-language verification. The current model was trained and evaluated exclusively on English Latin-based signatures, extending the framework to logographic [22] and Brahmic scripts as well as other languages in the Latin scripts [23][24][25] would provide deeper insights into its adaptability. Although CEDAR provided a controlled environment for writer-independent evaluation, future work could involve multilingual training datasets or domain adaptation strategies to improve cross-dataset generalisation.

Additionally, while the collection of larger and more diverse datasets was recommended, it was noted that simply increasing the number of signers per dataset does not necessarily improve discriminative ability within a class; rather, it simply enables the model to recognise a broader range of signature styles. Future endeavours in this domain should prioritise increasing the number of skilled forgeries per signer to develop a more robust model applicable to real-world systems.

Additionally, further extensive hyperparameter optimisation might enhance performance, variables such as margin selection, embedding dimensionality, and batch composition ratios were selected based on empirical reasoning rather than exhaustive search. As prior empirical studies [26] suggested that gains from exhaustive tuning often diminish once key parameters fall within stable ranges, any future optimisation should be conducted selectively or structurally. Future work may conduct a systematic sensitivity analysis over the margin, μ, λ, batch size, and *PKFM* composition to quantify how each parameter affects convergence and verification performance.

From a practical perspective, adaptive threshold calibration should be explored. Since security-sensitive applications require different trade-offs between False Acceptance Rates (FAR) and False Rejection Rates (FRR). Investigating dynamic or writer-specific thresholding strategies could improve operational performance in diverse environments.

REFERENCES

[1] Z. Hashim, A. Salman, and A. Alkhayyat, “A Comparative Study among Handwritten Signature Verification Methods Using Machine Learning Techniques,” *Scientific Programming*, vol. 2022, Oct. 2022, doi: https://doi.org/10.1155/2022/8170424.

[2] H. Xuan, A. Stylianou, X. Liu, and R. Pless, “Hard Negative Examples are Hard, but Useful,” in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 126–142.

[3] M. Stauffer, P. Maergner, A. Fischer, and K. Riesen, “A Survey of State of the Art Methods Employed in the Offline Signature Verification Process,” in *New Trends in Business Information Systems and Technology: Digital Innovation and Digital Business Transformation*, R. Dornberger, Ed., Cham: Springer International Publishing, 2021, pp. 17–30. doi: https://doi.org/10.1007/9783030483326_2.

[4] J. Poddar, V. Parikh, and B. S. Kumar, “Offline Signature Recognition and Forgery Detection using Deep Learning,” *The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops*, vol. 170, pp. 610–617, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.133.

[5] M. Mutlu Yapici, A. Tekerek, and N. Topaloglu, “Convolutional Neural Network Based Offline Signature Verification Application,” in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 30–34. doi: https://doi.org/10.1109/IBIGDELFT.2018.8625290.

[6] S. Tehsin, A. Hassan, F. Riaz, I. M. Nasir, N. L. Fitriyani, and M. Syafrudin, “Enhancing Signature Verification Using Triplet Siamese Similarity Networks in Digital Documents,” *Mathematics*, vol. 12, no. 17, p. 2757, Sep. 2024, doi: https://doi.org/10.3390/math12172757.

[7] Q. Wan and Q. Zou, “Learning Metric Features for WriterIndependent Signature Verification using Dual Triplet Loss,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 3853–3859. doi: https://doi.org/10.1109/ICPR48806.2021.9413091.

[8] W. Xiao and Y. Ding, “A Two-Stage Siamese Network Model for Offline Handwritten Signature Verification,” *Symmetry*, vol. 14, no. 6, p. 1216, Jun. 2022, doi: https://doi.org/10.3390/sym14061216.

[9] L. Liu, L. Huang, F. Yin, and Y. Chen, “Offline signature verification using a region based deep metric learning network,” *Pattern Recognition*, vol. 118, p. 108009, May 2021, doi: https://doi.org/10.1016/j.patcog.2021.108009.

[10] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” IEEE, Jun. 2015, pp. 815–823. doi: https://doi.org/10.1109/cvpr.2015.7298682.

[11] M. Tan and Q. V. Le, “EfficientNetV2: Smaller models and faster training,” *arXiv.org*, 2021. https://arxiv.org/abs/2104.00298

[12] M. Saks, “Commentary on: Srihari SN, Cha SH, Arora H, Lee S. Individuality of handwriting. J Forensic Sci 2002; 47(4):85672,” *Journal of forensic sciences*, vol. 48, pp. 916–8; author reply 919, Aug. 2003, doi: https://doi.org/10.1520/JFS2002428.

[13] Z. Zhang, C. Luo, H. Wu, Y. Chen, N. Wang, and C. Song, “From Individual to Whole: Reducing Intraclass Variance by Feature Aggregation,” *International Journal of Computer Vision*, vol. 130, no. 3, pp. 800–819, 2022, doi: https://doi.org/10.1007/s11263021015692.

[14] M. K. KALERA, S. SRIHARI, and A. XU, “OFFLINE SIGNATURE VERIFICATION AND IDENTIFICATION USING DISTANCE STATISTICS,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 07, pp. 1339–1360, Nov. 2004, doi: https://doi.org/10.1142/s0218001404003630.

[15] “CEDAR Signature Verification,” *Buffalo.edu*, 2025. https://cedar.buffalo.edu/signature/ (accessed Nov. 28, 2025).

[16] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed. “O’Reilly Media, Inc.,” 2022, pp. 391–715.

[17] A. Vaswani *et al.*, “Attention is all you need,” *arXiv.org*, 2023. https://arxiv.org/abs/1706.03762

[18] B. Harwood, G. Vijay, G. Carneiro, I. Reid, and T. Drummond, “Smart Mining for Deep Metric Learning,” *arXiv.org*, 2017. https://arxiv.org/abs/1704.01285 (accessed May 02, 2026).

[19] M. Kaya and H. Ş. Bilge, “Deep Metric Learning: A Survey,” *Symmetry*, vol. 11, no. 9, p. 1066, Aug. 2019, doi: https://doi.org/10.3390/sym11091066.

[20] R. Geirhos *et al.*, “Shortcut Learning in Deep Neural Networks,” *arXiv (Cornell University)*, vol. 2, no. 11, Apr. 2020, doi: https://doi.org/10.48550/arxiv.2004.07780.

[21] M. A. Islam, S. Jia, and B. Neil, “How Much Position Information Do Convolutional Neural Networks Encode?,” *arXiv.org*, 2020. https://arxiv.org/abs/2001.08248v1 (accessed Apr. 16, 2026).

[22] hsinmin, “GitHub - hsinmin/HanSig: A large-scale offline Chinese handwritten signature dataset,” *GitHub*, 2025. https://github.com/hsinmin/HanSig (accessed Nov. 28, 2025).

[23] F. Vargas, M. Ferrer, C. Travieso, and J. Alonso, “Offline Handwritten Signature GPDS960 Corpus,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, pp. 764–768. doi: https://doi.org/10.1109/ICDAR.2007.4377018.

[24] F.-H. Huang and H.-M. Lu, “Multiscale feature learning using co-tuplet loss for offline handwritten signature verification,” *arXiv.org*, 2025. https://arxiv.org/abs/2308.00428

[25] OrtegaGarcia J *et al.*, “MCYT baseline corpus: a bimodal biometric database,” *IEE ProceedingsVision, Image and Signal Processing*, vol. 150, no. 6, pp. 395–401, 2003, doi: https://doi.org/10.1049/ipvis:20031078.

[26] M. Sipper, “High per parameter: A large-scale study of hyperparameter tuning for machine learning algorithms,” *Algorithms*, vol. 15, Art. no. 9, 2022, doi: https://doi.org/10.3390/a15090315.

Appendix A

Signatures Collected

Signer 1 Genuine Signatures

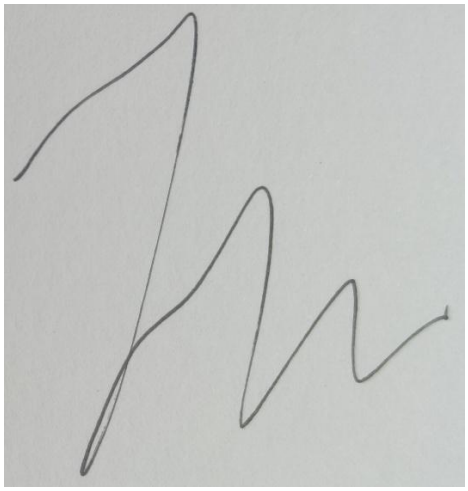


Figure A1 Signer 1 Genuine 1

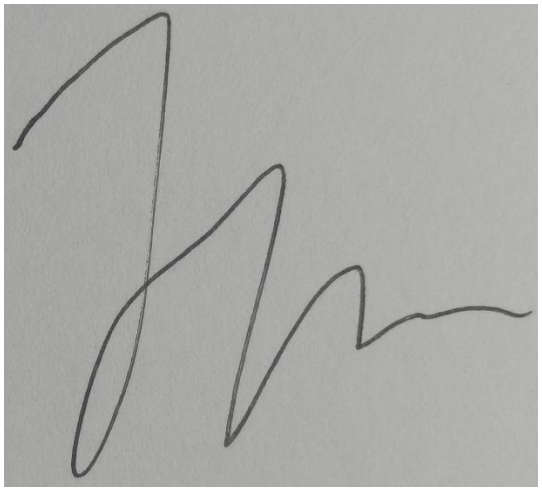


Figure A2 Signer 1 Genuine 2

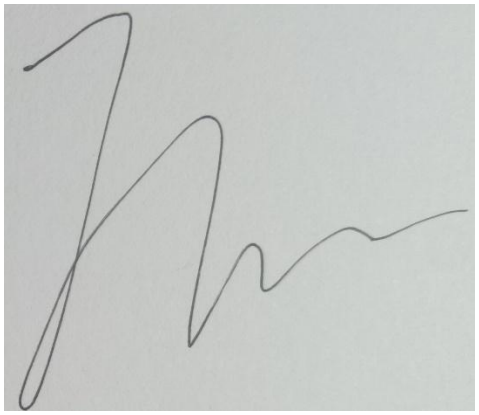


Figure A3 Signer 1 Genuine 3

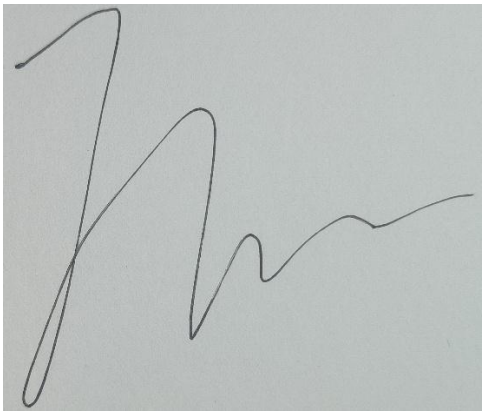


Figure A4 Signer 1 Genuine 4

Signer 1 Forged Signatures

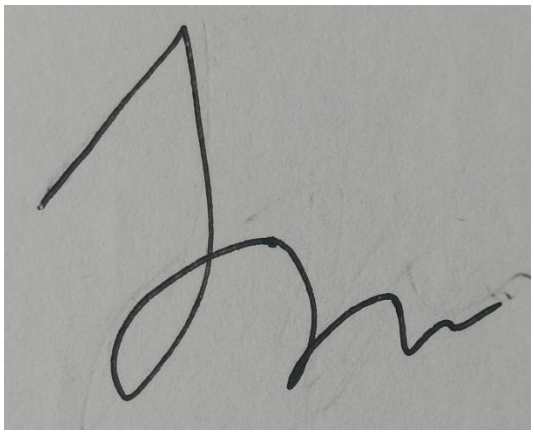


Figure A5 Signer 1 Forged 1

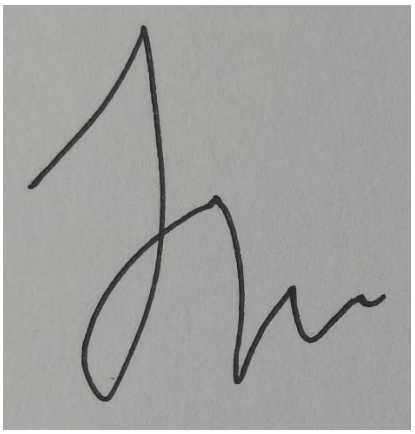


Figure A6 Signer 1 Forged 2

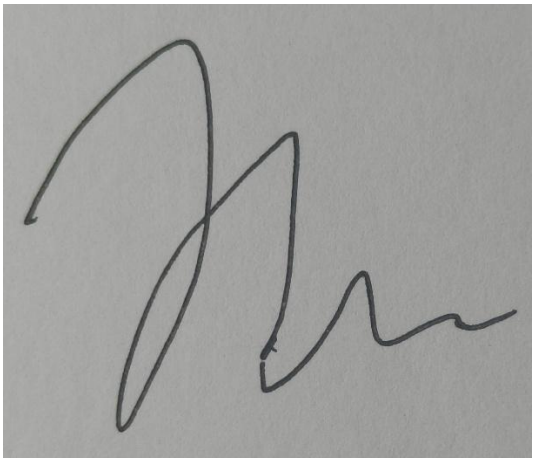


Figure A7 Signer 1 Forged 3

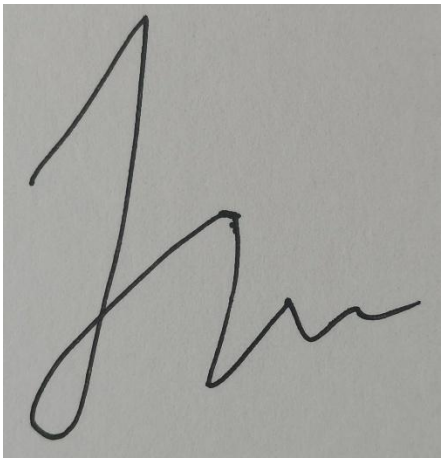


Figure A8 Signer 1 Forged 4

Signer 2 Genuine Signatures

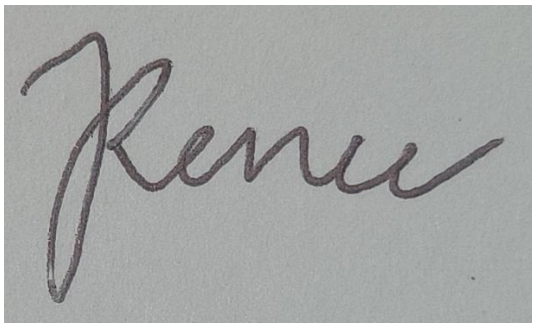


Figure A9 Signer 2 Genuine 1

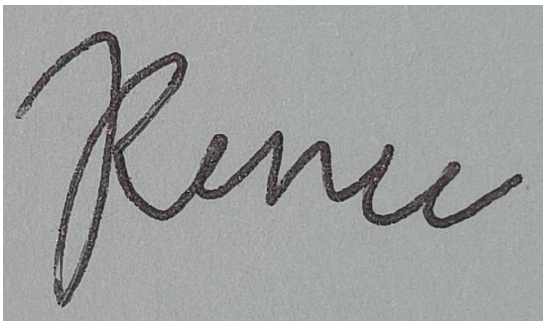


Figure A10 Signer 2 Genuine 2

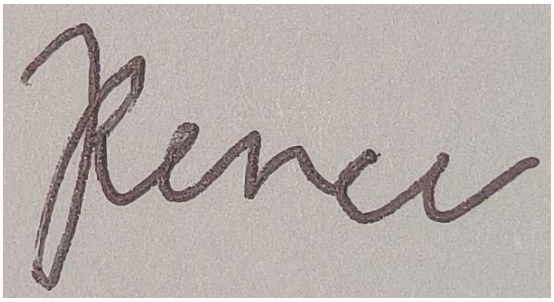


Figure A11 Signer 2 Genuine 3

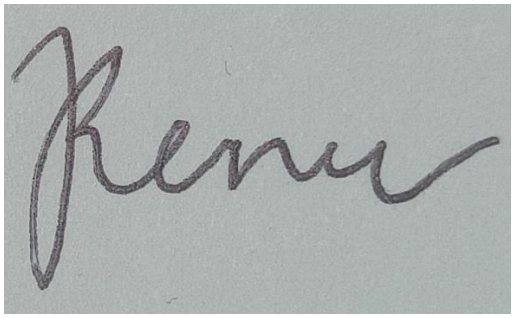


Figure A12 Signer 2 Genuine 4

Signer 2 Forged Signatures

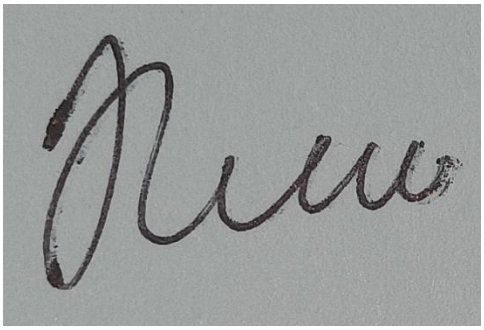


Figure A13 Signer 2 Forged 1

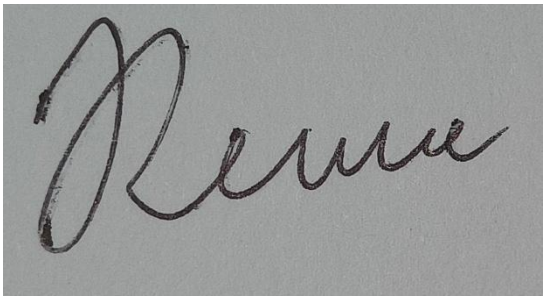


Figure A14 Signer 2 Forged 2

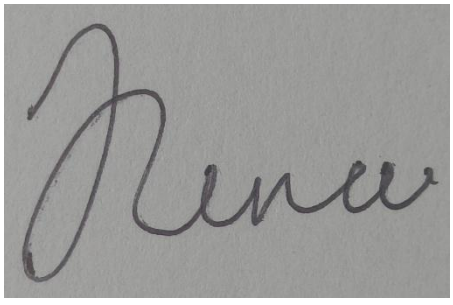


Figure A15 Signer 2 Forged 3

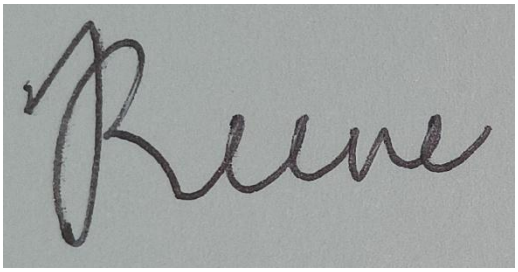


Figure A16 Signer 2 Forged 4

Signer 3 Genuine Signatures

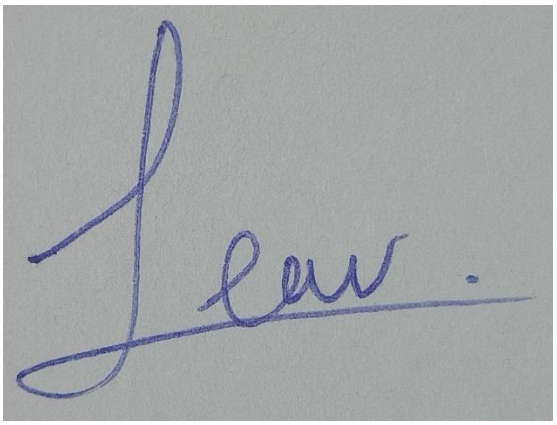


Figure A17 Signer 3 Genuine 1

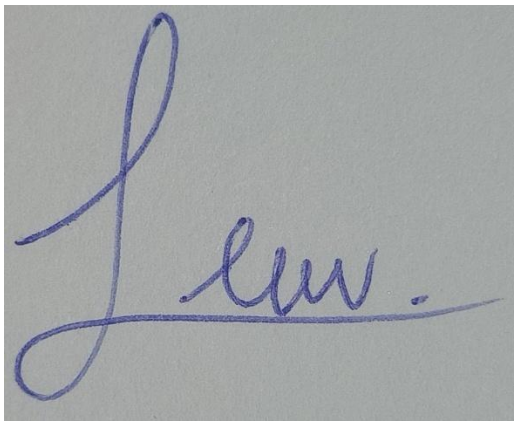


Figure A18 Signer 3 Genuine 2

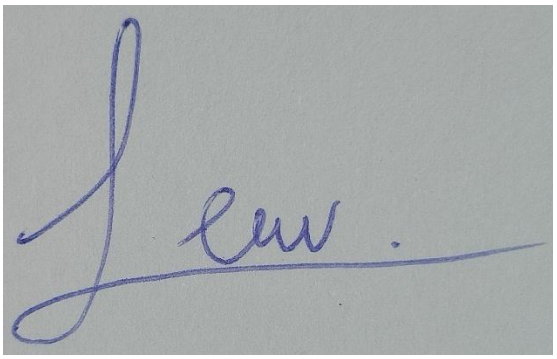


Figure A19 Signer 3 Genuine 3

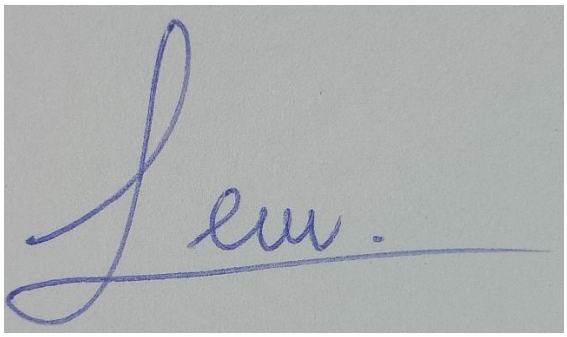


Figure A20 Signer 3 Genuine 4

Signer 3 Forged Signatures

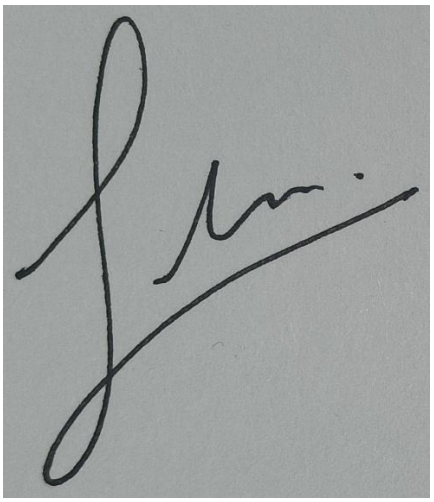


Figure A21 Signer 3 Forged 1

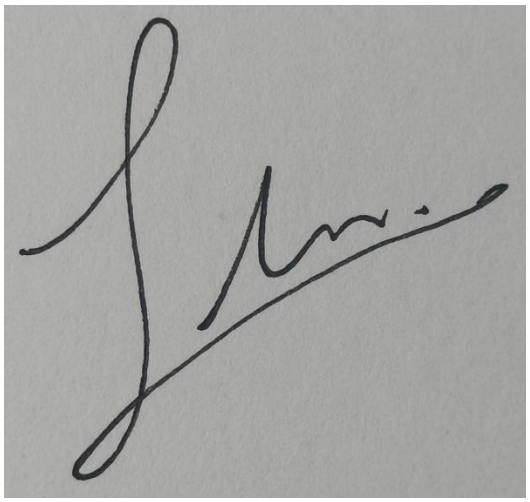


Figure A22 Signer 3 Forged 2

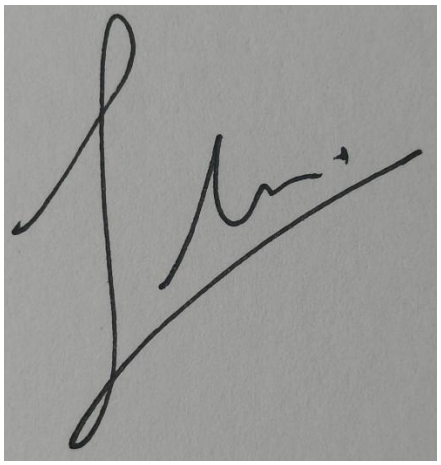


Figure A23 Signer 3 Forged 3

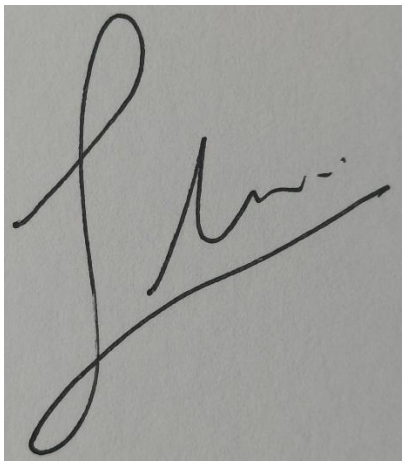


Figure A24 Signer 3 Forged 4

Signer 4 Genuine Signatures

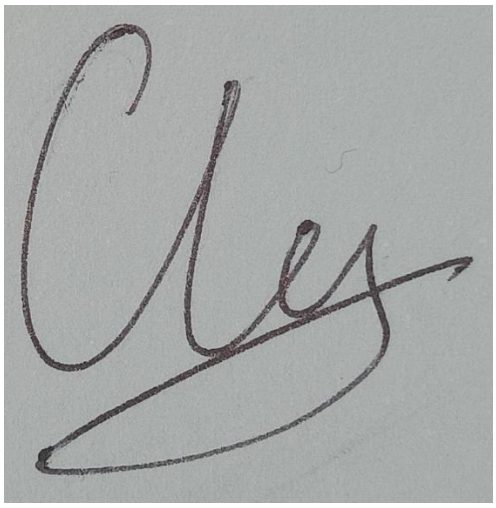


Figure A25 Signer 4 Genuine 1

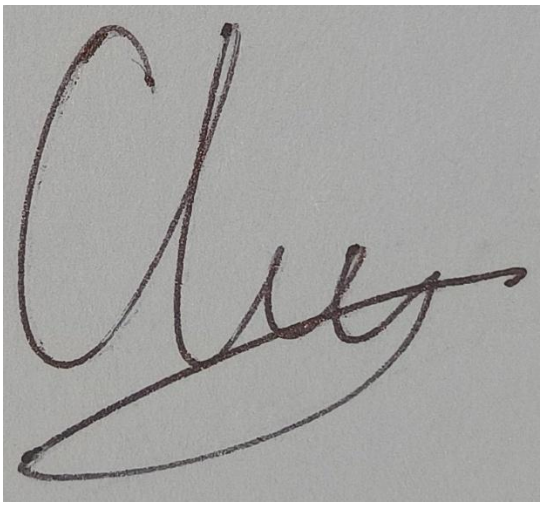


Figure A26 Signer 4 Genuine 2

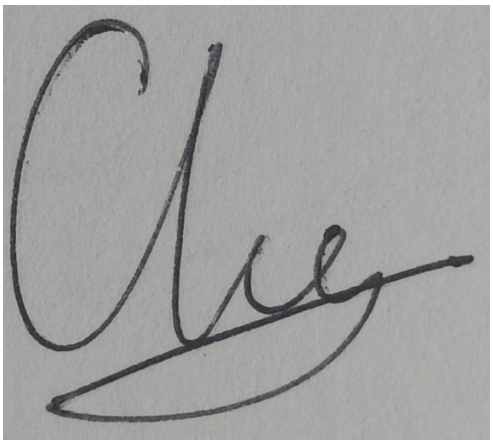


Figure A27 Signer 4 Genuine 3

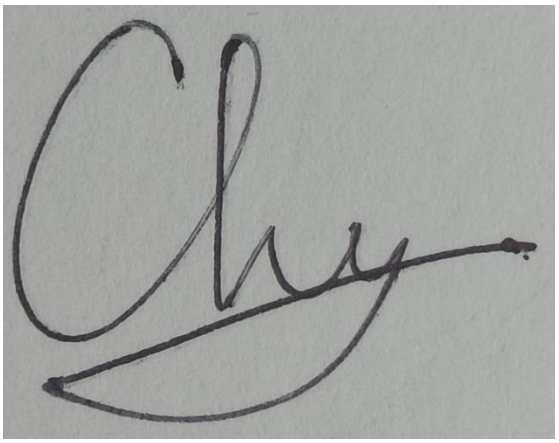


Figure A28 Signer 4 Genuine 4

Signer 4 Forged Signatures

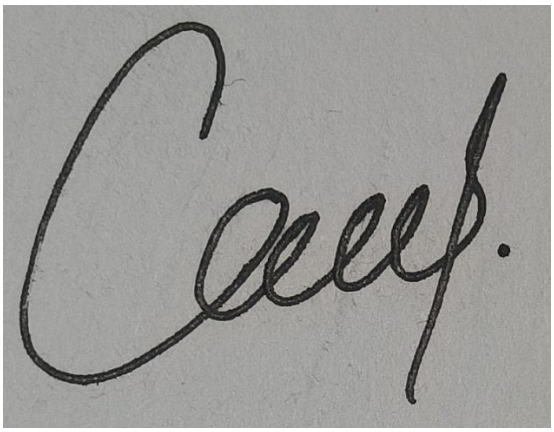


Figure A29 Signer 4 Forged 1

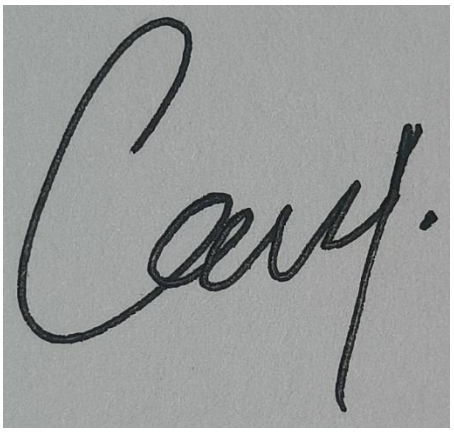


Figure A30 Signer 4 Forged 2

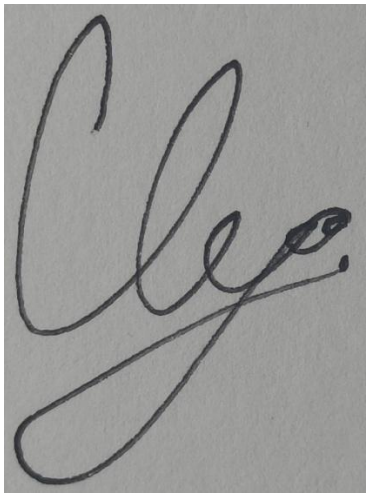


Figure A31 Signer 4 Forged 3

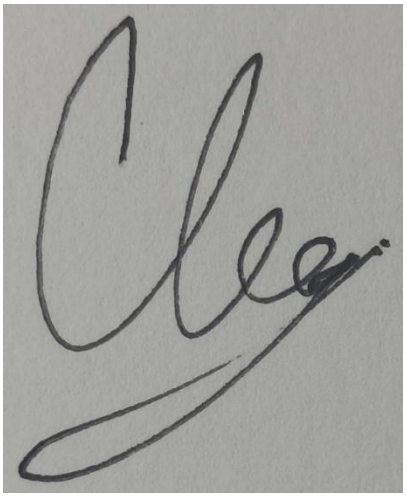


Figure A32 Signer 4 Forged 4

Signer 5 Genuine Signatures

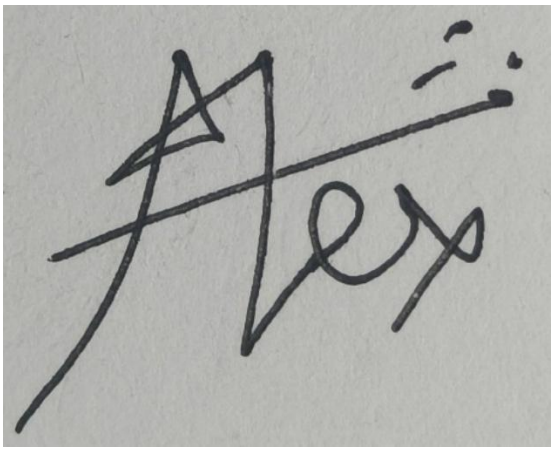


Figure A33 Signer 5 Genuine 1

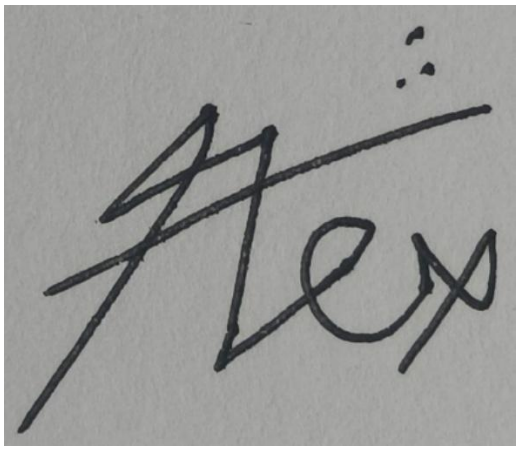


Figure A34 Signer 5 Genuine 2

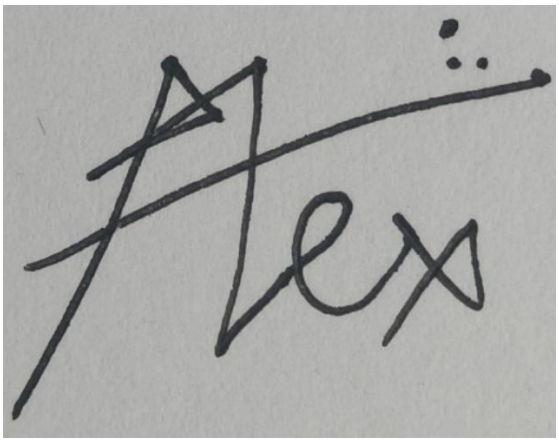


Figure A35 Signer 5 Genuine 3

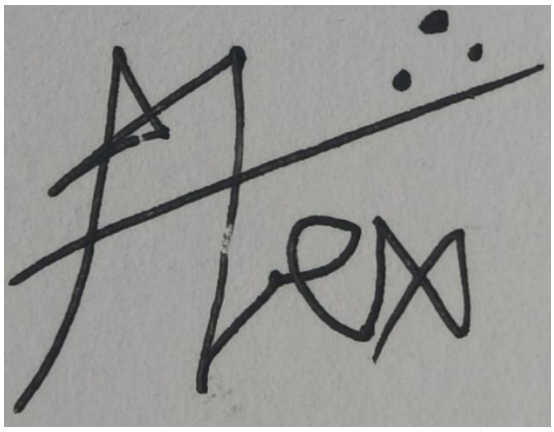


Figure A36 Signer 5 Genuine 4

Signer 5 Forged Signatures

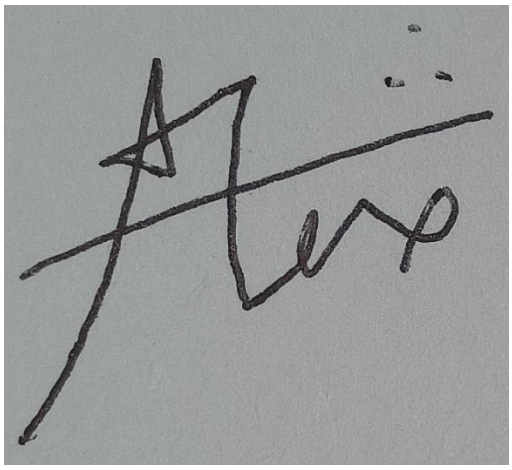


Figure A35 Signer 5 Forged 1

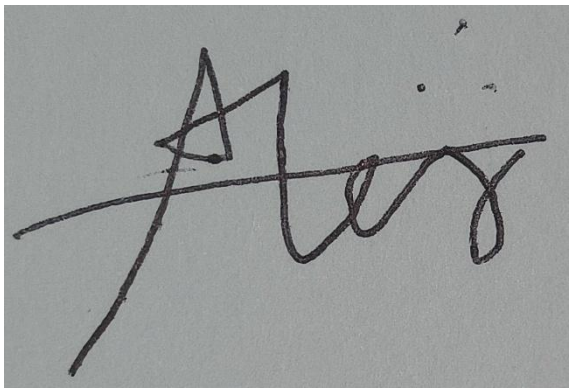


Figure A36 Signer 5 Forged 2

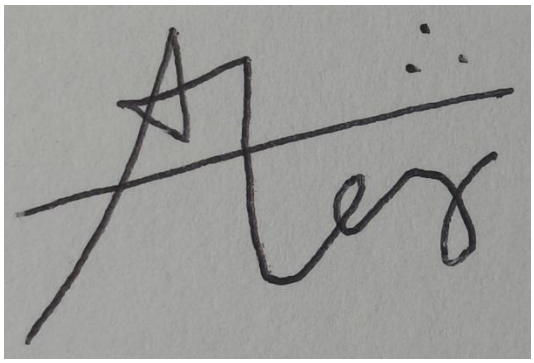


Figure A37 Signer 5 Forged 3

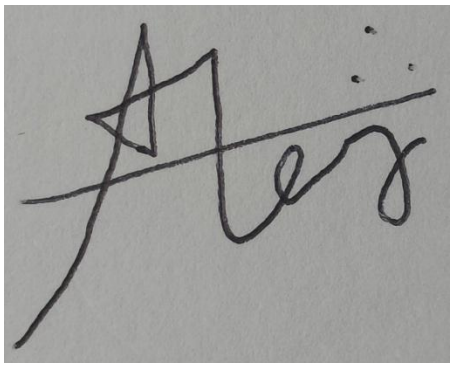


Figure A38 Signer 5 Forged 4

Appendix B

False Positive Pairs (Grad-CAM)

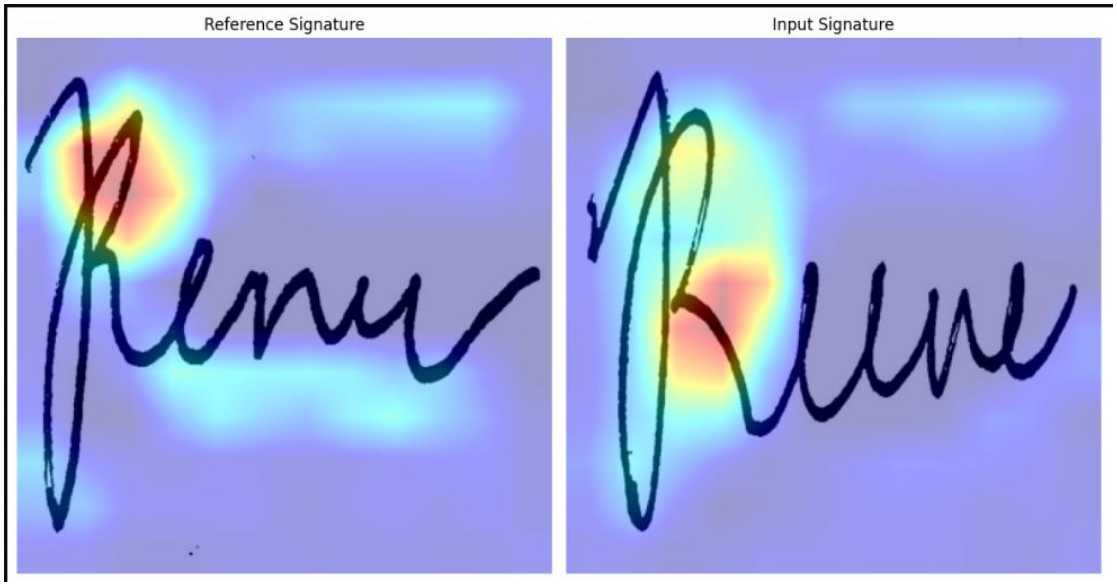


Figure B1 Pair 1

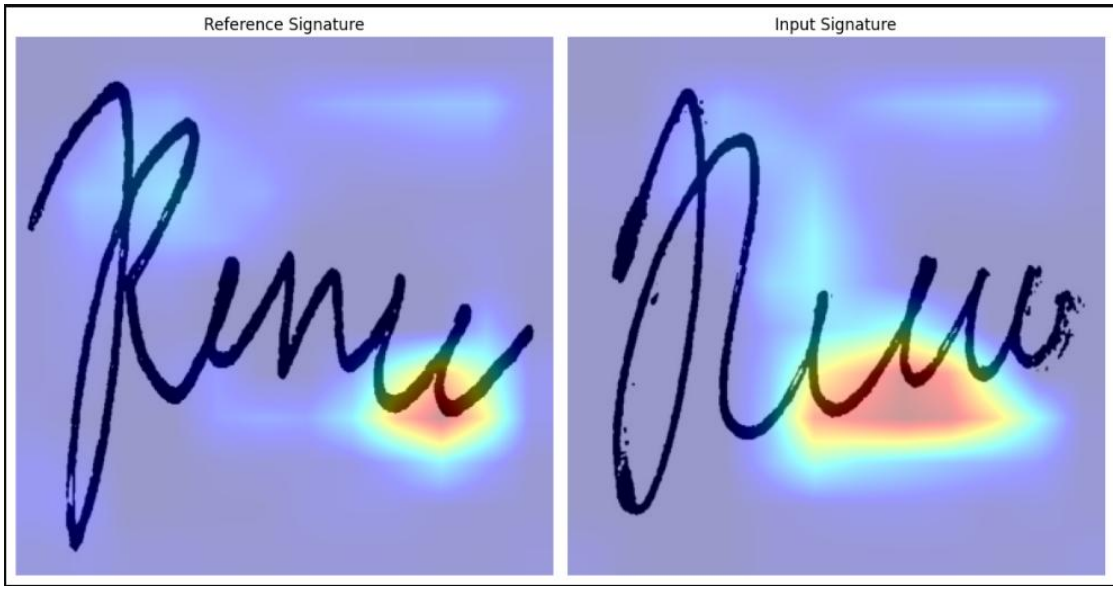


Figure B2 Pair 2

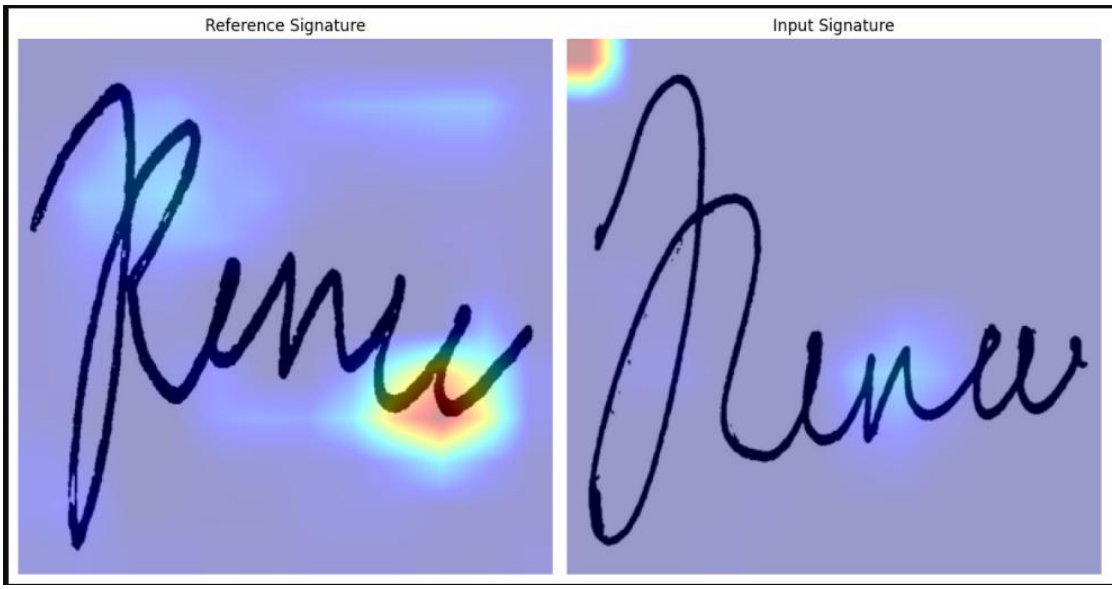


Figure B3 Pair 3

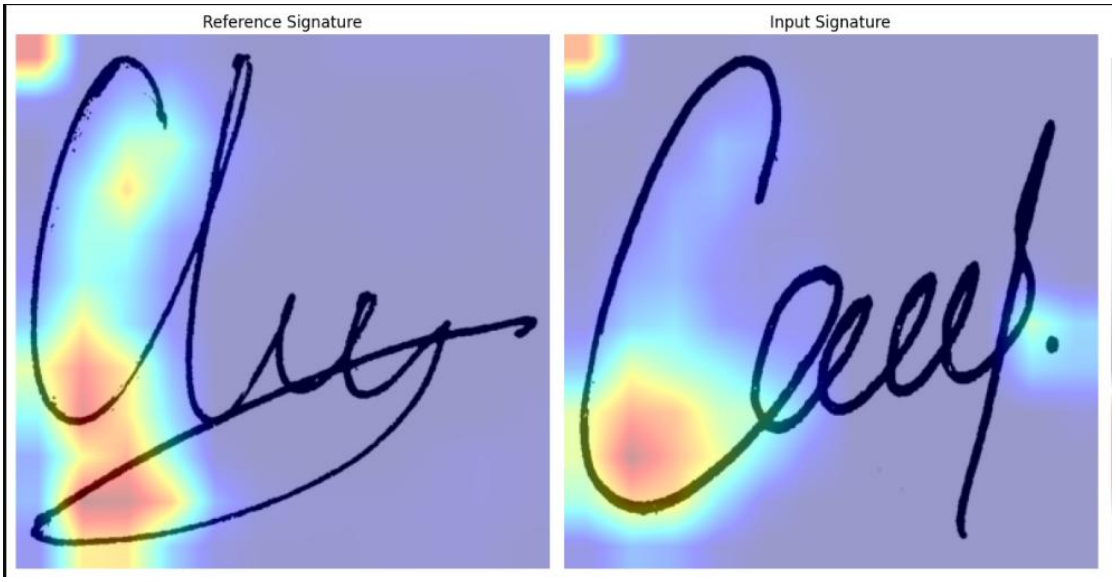


Figure B4 Pair 4

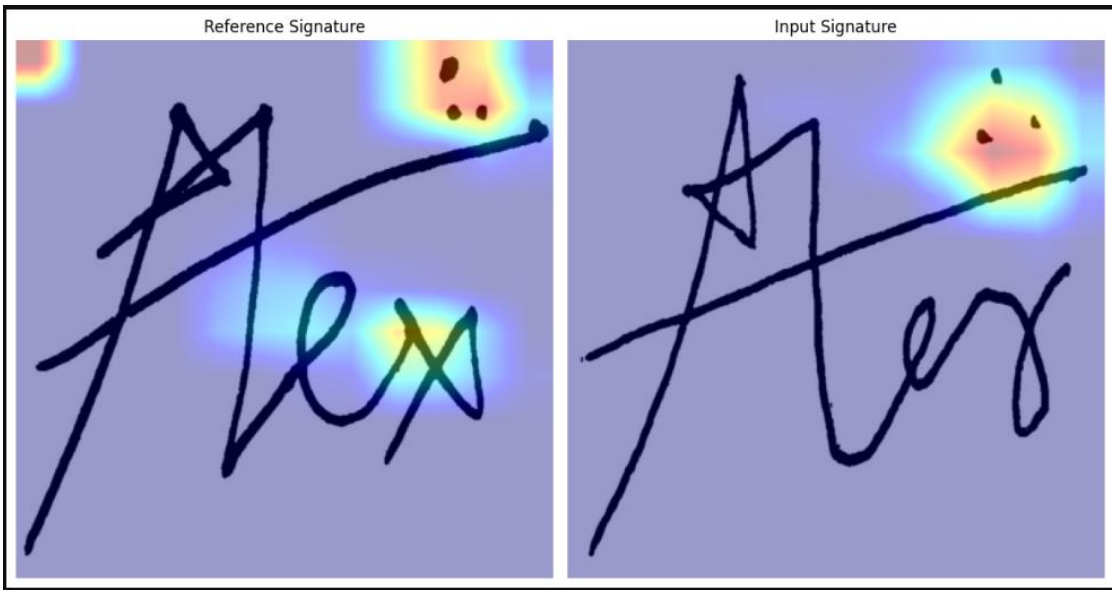


Figure B5 Pair 5

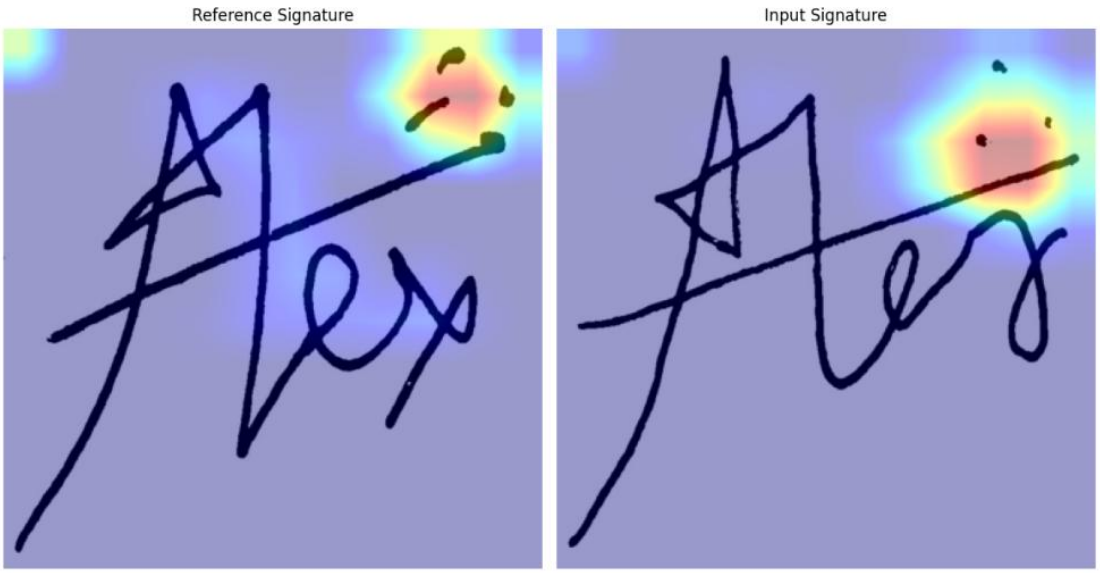


Figure B6 Pair 6

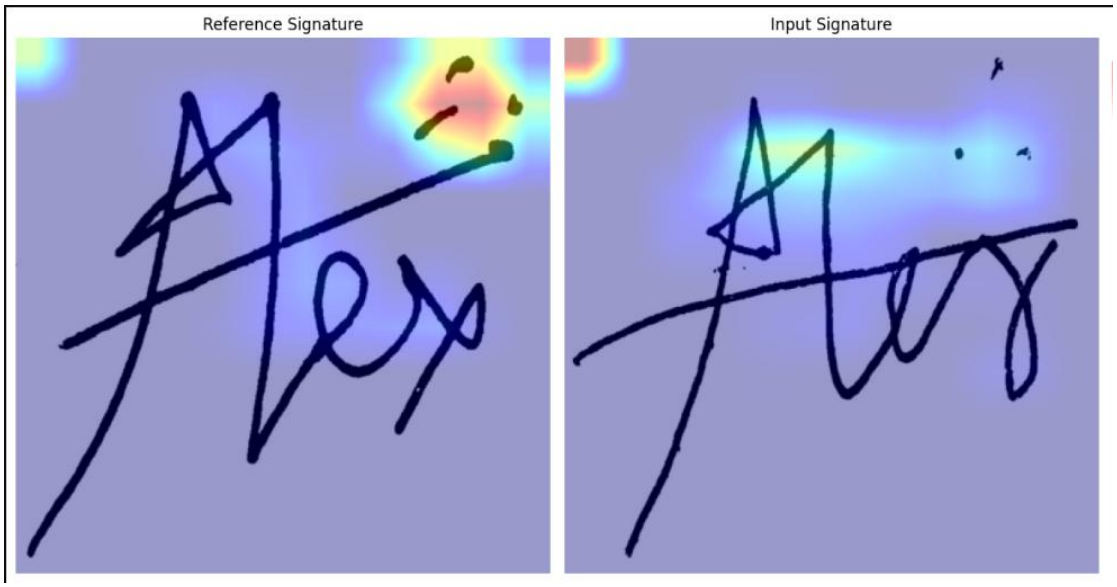


Figure B7 Pair 7

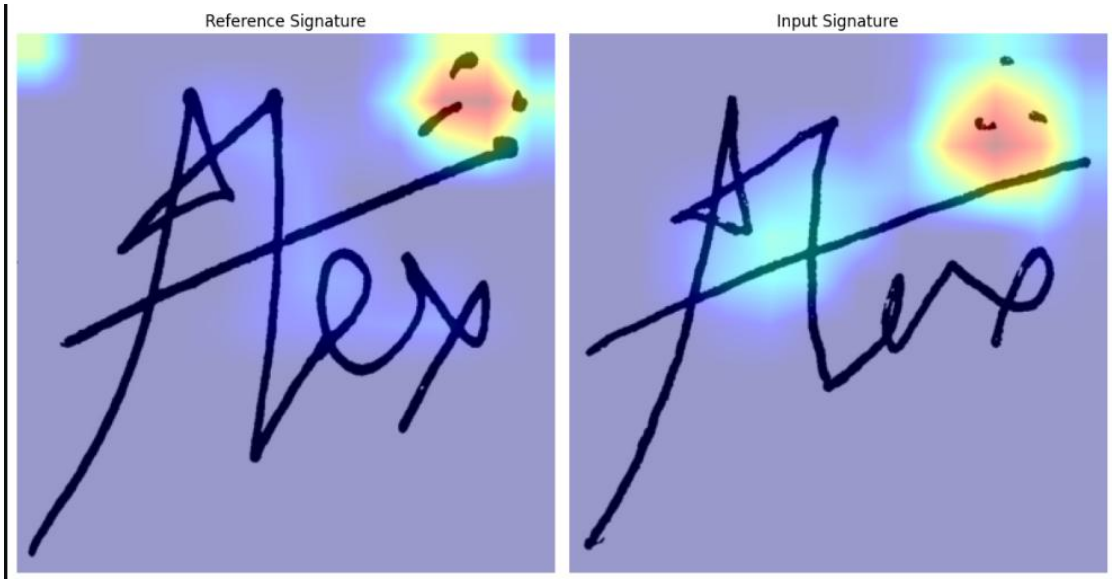


Figure B8 Pair 8

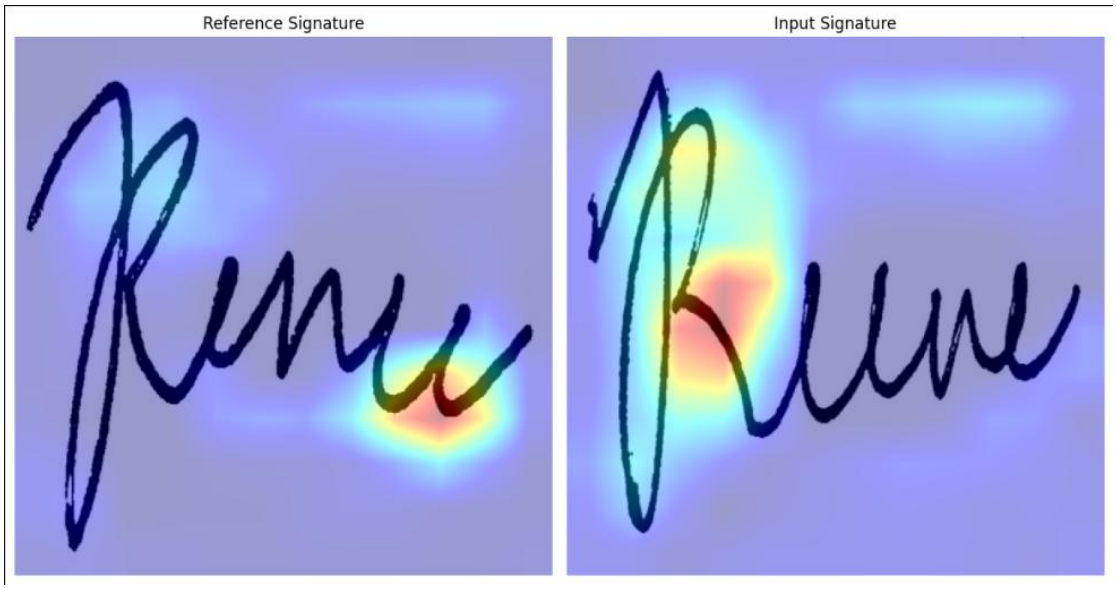


Figure B9 Pair 9

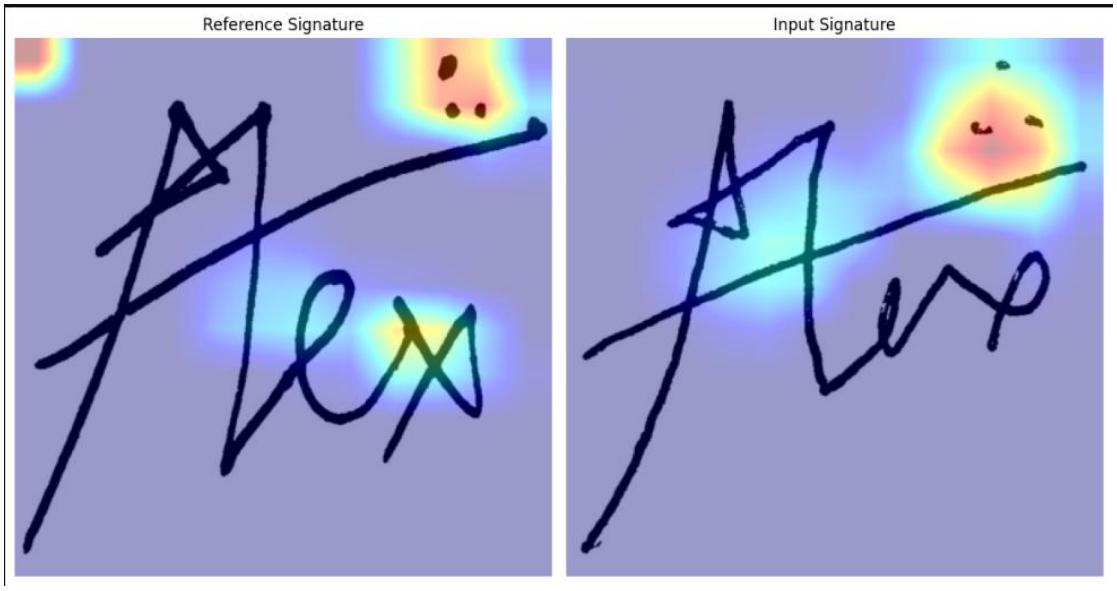


Figure B10 Pair 10

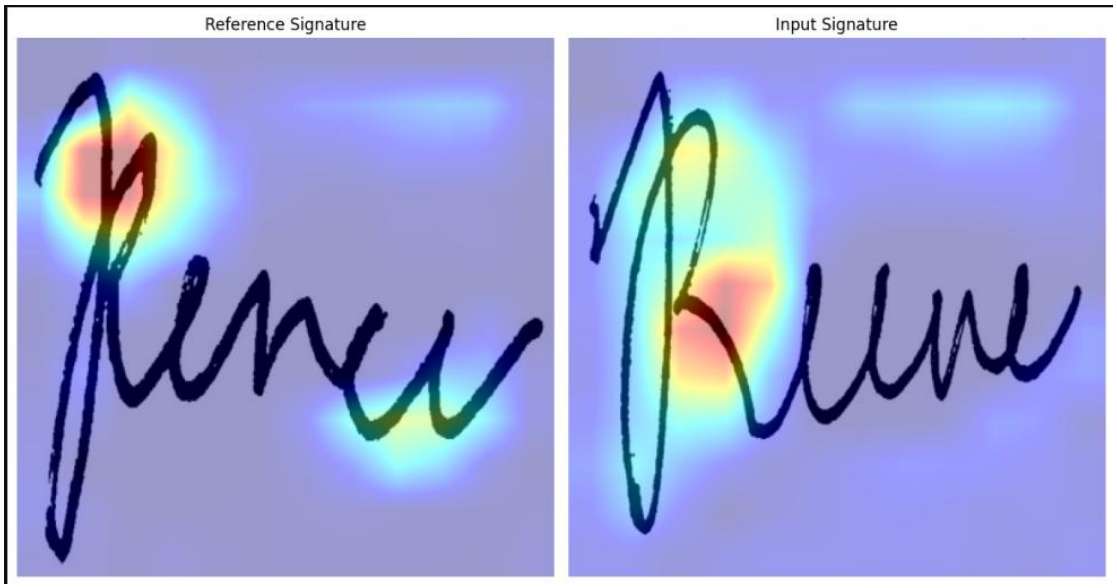


Figure B11 Pair 11

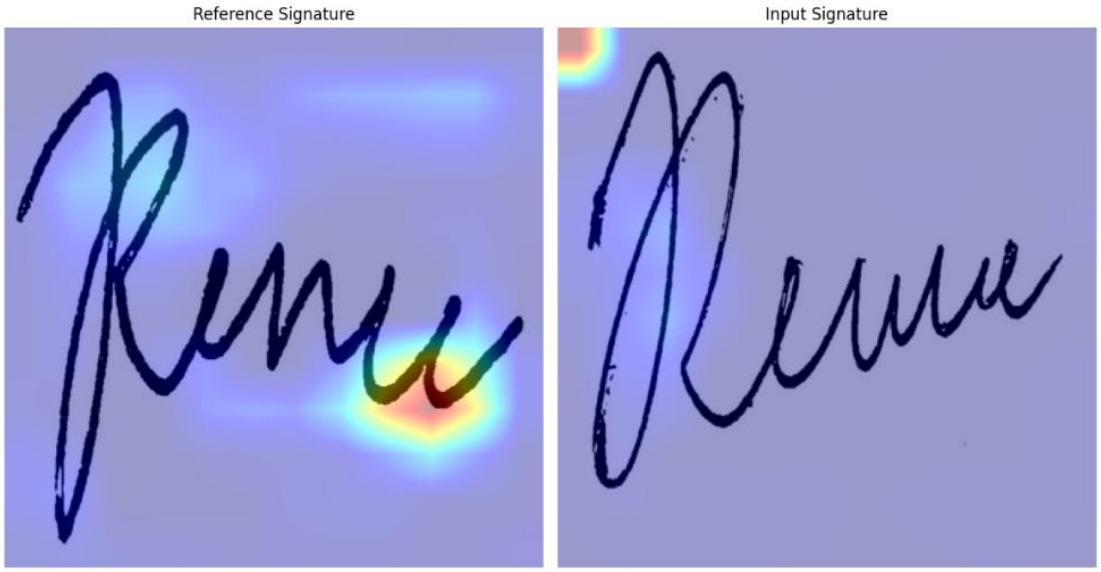


Figure B12 Pair 12

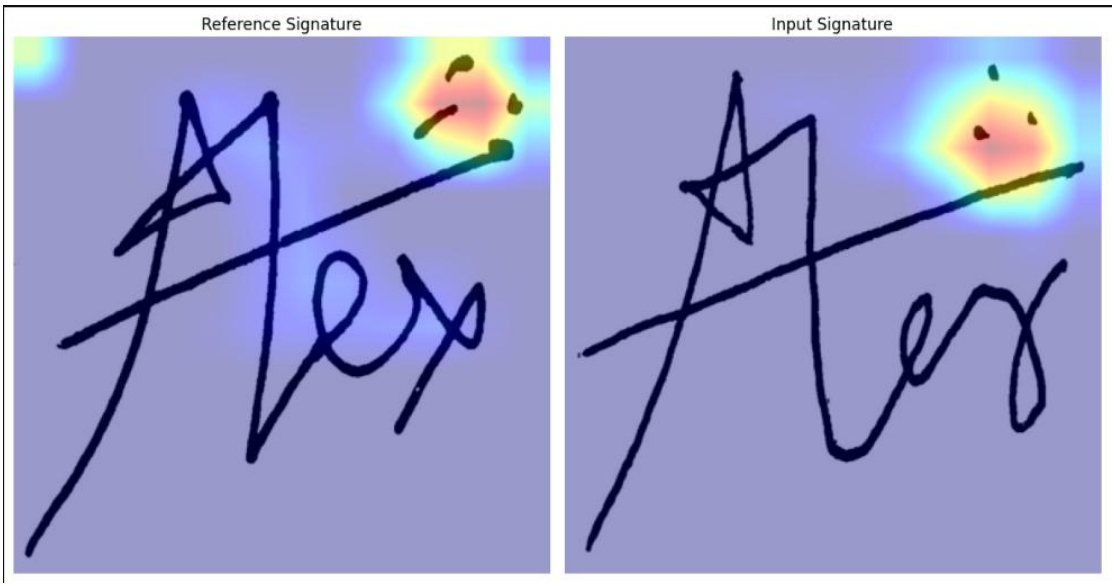
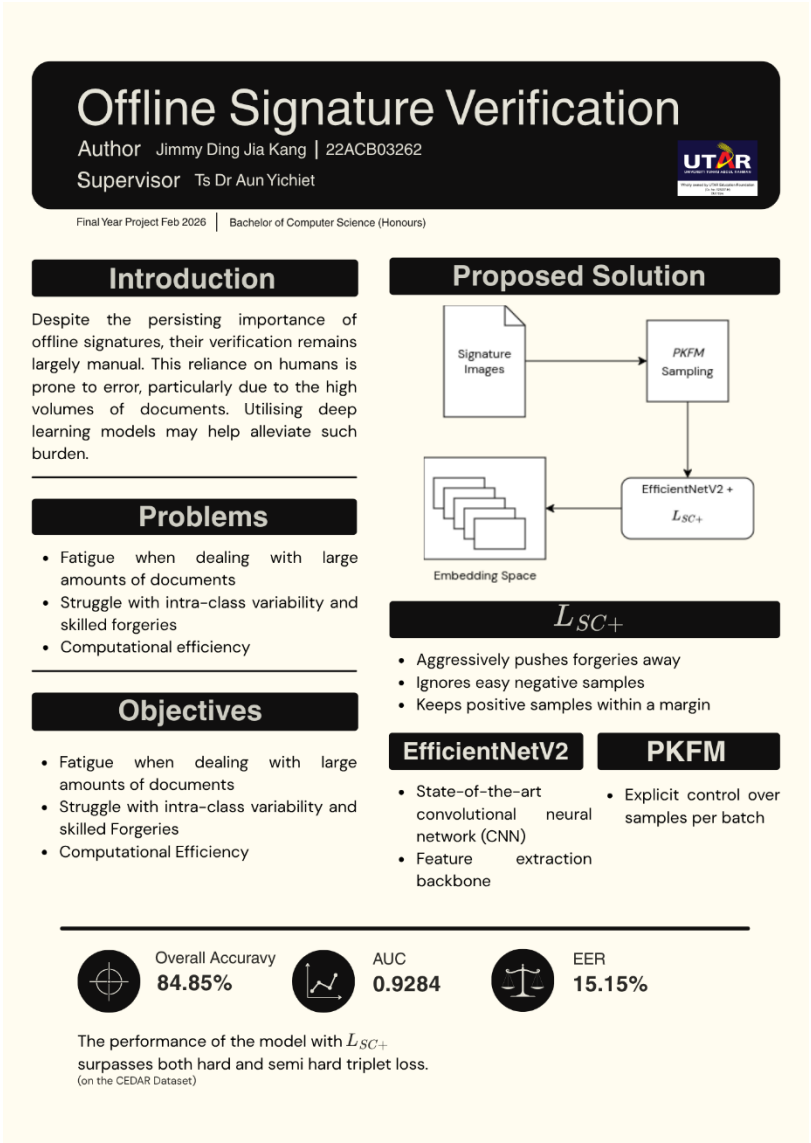


Figure B13 Pair 13

POSTER


Offline Signature Verification
Author Jimmy Ding Jia Kang | 22ACB03262
Supervisor Ts Dr Aun Yichiet
Final Year Project Feb 2026 | Bachelor of Computer Science (Honours)
Introduction
Despite the persisting importance of offline signatures, their verification remains largely manual. This reliance on humans is prone to error, particularly due to the high volumes of documents. Utilising deep learning models may help alleviate such burden.
Problems
• Fatigue when dealing with large amounts of documents
• Struggle with intra-class variability and skilled forgeries
• Computational efficiency
Objectives
• Fatigue when dealing with large amounts of documents
• Struggle with intra-class variability and skilled Forgeries
• Computational Efficiency
Proposed Solution
Signature Images
PKFM Sampling
EfficientNetV2 + L_{SC+}
Embedding Space
L_{SC+}
• Aggressively pushes forgeries away
• Ignores easy negative samples
• Keeps positive samples within a margin
EfficientNetV2
• State-of-the-art convolutional neural network (CNN)
• Feature extraction backbone
PKFM
• Explicit control over samples per batch
Overall Accuravy 84.85%
AUC 0.9284
EER 15.15%
The performance of the model with L_{SC+} surpasses both hard and semi hard triplet loss.
(on the CEDAR Dataset)