

Minor Project Report

on

SIGNATURE AUTHENTICATION USING SIAMESE NEURAL NETWORK

Submitted in partial fulfillment of the
requirements for the completion of Minor Project [ARP 455]

Name: **Ankit Tayal**

Enrollment No.: **04319011721**

Under the supervision of

DR. KHYATI CHOPRA



**UNIVERSITY SCHOOL OF AUTOMATION AND ROBOTICS
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
EAST DELHICAMPUS, SURAJMAL VIHAR, DELHI- 110092**

DECLARATION

I hereby declare that the Minor Project Report entitled **Predictive Models for Signature Authentication Using Siamese Neural Network** is an authentic record of work completed as requirements of Minor Project (ARP 455) during the period from **10/09/2024 to 15/11/2024** in University School of Automation and Robotics under the supervision of **Dr. Khyati Chopra**

Name: Ankit Tayal

Enrollment No: 04319011721

Date: _____

Dr. Khyati Chopra

Date: _____

I

ACKNOWLEDGEMENT

We would like to take the opportunity to thank and express our deep sense of gratitude to our mentor and project guide **Dr. Khyati Chopra** for his immense support and valuable guidance without which it would not have been possible to reach at this stage of our minor project.

I also obliged to all our faculty members for their valuable support in their respective fields which helped us in reaching at this stage of our project.

I extend my appreciation to the entire University School of Automation and Robotics, where I was given the platform to apply theoretical knowledge to real-world scenarios. I am grateful to the staff and fellow interns for creating a collaborative and enriching atmosphere.

II

ABOUT INSTITUTE

Guru Gobind Singh Indraprastha University (GGSIPU) is the first University established in 1998 by Govt. of NCT of Delhi under the provisions of Guru Gobind Singh Indraprastha University Act, 1998 read with its Amendment in 1999. The University is recognized by University Grants Commission (UGC), India under section 12B of UGC Act.

The Guru Gobind Singh Indraprastha University has been Accredited with a CGPA of 3.56 on a seven-point scale at A++ Grade valid for a period of 7 years from 14-02-2024.

It is a teaching and affiliating University with the explicit objective of facilitating and promoting “studies, research and extension work in emerging areas of higher education with focus on professional education, for example engineering, technology, management studies, medicine, pharmacy, nursing, education, law, etc. and also to achieve excellence in these and connected fields and other matters connected therewith or incidental thereto.”

In order to serve the broad purposes for which the University was established, it set out its statements of Vision, Mission and Quality Policy which read as thus:

VISION

“The University will stimulate both the hearts and minds of scholars, empower them to contribute to the welfare of society at large; train them to adopt themselves to the changing needs of the economy; advocate them for cultural leadership to ensure peace, harmony and prosperity for all

III

TABLE OF CONTENTS

- 1. Chapter 1: ABSTRACT**
 - 1.1. Abstract**
- 2. Chapter 2: INTRODUCTION**
 - 2.1. General Introduction**
 - 2.2. Importance of Signature Authentication**
 - 2.3. Role of Deep Learning in Signature Verification**
- 3. Chapter 3: LITERATURE REVIEW**
 - 3.1. Overview of Traditional Signature Authentication Methods**
 - 3.2. Machine Learning Techniques for Signature Verification**
 - 3.3. Siamese Neural Networks in Biometric Systems**
- 4. Chapter 4: METHODOLOGY**
 - 4.1. Project Scope and Objectives**
 - 4.2. Data Collection and Datasets (e.g., BHSig260, Cedar)**
 - 4.3. Data Preprocessing**
 - 4.4. Model Architecture: Siamese Neural Network**
 - 4.5. Training Configuration and Hyperparameter Tuning**
- 5. Chapter 5: IMPLEMENTATION**
 - 5.1. Technology Stack and Tools Used**
 - 5.2. Model Implementation Details**
 - 5.3. Optimization Techniques**

6. Chapter 6: RESULTS AND DISCUSSION

6.1. Model Performance Metrics (Accuracy, Precision, Recall, F1-score)

6.2. Comparative Analysis with Baseline Models

6.3. Visualizations (ROC Curve, Confusion Matrix)

6.4. Interpretation of Results

7. Chapter 7: CONCLUSION AND FUTURE WORK

7.1. Challenges Faced

7.2. Model Limitations

7.3. Future Enhancements

7.4. Summary and Conclusion

8. Chapter 8: REFERENCES

IV

LIST OF ABBREVIATIONS

Abbreviation	Full Form
DL	Deep Learning
CNN	Convolutional Neural Network
SNN	Siamese Neural Network
BHSig260	Bengali and Hindi Signature Dataset 260
Cedar	Cedar Handwritten Signature Dataset
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
HDF5	Hierarchical Data Format 5
GPU	Graphics Processing Unit
ReLU	Rectified Linear Unit
LR	Learning Rate
Adam	Adaptive Moment Estimation
RMSprop	Root Mean Square Propagation
KNN	K-Nearest Neighbors
ML	Machine Learning
AI	Artificial Intelligence
TF	TensorFlow
Keras	Keras API Framework
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
F1-Score	Harmonic Mean of Precision and Recall

LIST OF FIGURES

FIGURE 1 FEATURE SHAPE	20
FIGURE 2 MODEL 1 CEDAR WITH 1000 EPOCH	22
FIGURE 3 GOOGLE COLAB WITH GPU 334GB RAM	24
FIGURE 4 GOOGLE DRIVE PREPROCESSED DATA FILES.....	25
FIGURE 5 PLOT MODEL 2 (SIGNET)	27
FIGURE 6 GENUINE AND FORGED SIGNATURES	28
FIGURE 7 GENUINE AND FORGED SIGNATURES	30
FIGURE 8 COMPARISON ACROSS ALL MODELS	32
FIGURE 9 SCORE PREDICTION AND CLASSIFICATION	34
FIGURE 10 ROC CURVE COMPARISON ACROSS MODELS.....	35
FIGURE 11 CONFUSION MATRIX FOR MODEL5_RESNET18SIAMESE (BHSIG260 HINDI)	35
FIGURE 12 TRAINING AND VALIDATION LOSS FOR MODEL6_EFFICIENTNETB0SIAMESE	36
FIGURE 13 COMPARISON OF METRICS ACROSS ACCURACY BAR GRAPH	37

CHAPTER 1: ABSTRACT

In today's digital world, the need for secure and reliable authentication systems has grown significantly, particularly in areas like banking, legal documentation, and secure digital transactions. This project focuses on the development of a robust **signature authentication system** using **Siamese Neural Networks (SNNs)**, a specialized deep learning architecture designed for similarity-based tasks. The system aims to differentiate genuine signatures from forgeries by comparing pairs of images and analyzing their similarity through a learned feature space.

The methodology employed in this project begins with the collection and preprocessing of publicly available datasets, such as **BHSig260** (Bengali and Hindi signatures) and **Cedar**. Preprocessing steps include resizing, normalization, grayscale conversion, and data augmentation to improve model generalization. A Siamese Network architecture is implemented, utilizing shared weights to extract identical feature representations for both input signatures. The model's training process is optimized through hyperparameter tuning, and performance is evaluated using metrics like accuracy, precision, recall, F1-score, and AUC (Area Under Curve). Visualizations such as ROC curves and confusion matrices are used to interpret results effectively.

This research advances the development of **automated signature verification systems**, which offer critical applications in preventing fraud, enhancing document security, and streamlining verification processes. The project addresses challenges like class imbalance, variability in signature styles, and model overfitting by leveraging efficient training techniques and advanced neural network architectures.

Future research will focus on refining the Siamese Network model by incorporating state-of-the-art architectures like ResNet and EfficientNet, addressing issues such as real-world deployment, scalability, and performance under challenging conditions like occlusions and noise. This work not only contributes to the field of biometric authentication but also paves the way for robust and scalable solutions in signature verification systems for practical applications across various industries.

CHAPTER 2: INTRODUCTION

2.1 General Introduction

In today's increasingly digital world, the demand for secure and reliable authentication systems has never been more critical. From financial transactions to legal agreements, signatures remain a cornerstone of verification processes, symbolizing consent and authenticity. However, traditional methods of signature authentication, such as manual verification, are fraught with challenges, including human error, forgery risks, and inefficiencies. These limitations underline the need for automated solutions capable of addressing the growing complexities of modern authentication requirements.

The advent of deep learning has opened up new possibilities in the field of signature verification. Among various architectures, Siamese Neural Networks (SNNs) have emerged as a particularly effective tool for similarity-based tasks like signature authentication. Unlike conventional classifiers, SNNs specialize in comparing pairs of inputs to determine their similarity. This approach makes them uniquely suited for problems where relative comparison, rather than absolute classification, is key.

This project explores the design and implementation of a robust **Signature Authentication System** using SNNs. The system is trained to distinguish between genuine and forged signatures by analyzing their structural and visual features. Leveraging datasets such as **BHSig260** (Bengali and Hindi) and **Cedar**, the system employs advanced preprocessing techniques, feature extraction, and model optimization to ensure high accuracy and reliability. The outcomes of this project aim to pave the way for scalable solutions in industries where secure authentication is paramount.

2.2 Importance of Signature Authentication

Signatures hold a unique position in the realm of identity verification. Whether on a legal contract, a bank check, or a digital document, a signature serves as a personal identifier, validating the authenticity and intent of the signatory. However, as reliance on signatures grows, so does the risk of fraud. Forged signatures can lead to significant financial losses, reputational damage, and legal disputes.

Traditional methods of signature authentication, typically performed manually by trained experts, are often time-consuming and subjective. They rely heavily on human judgment, making them susceptible to inconsistency and error. Furthermore, the increasing digitization of documents necessitates solutions that can handle both handwritten and electronic signatures effectively.

An automated signature authentication system addresses these challenges by providing:

1. **Consistency:** Automated systems eliminate the subjectivity of human evaluation, ensuring uniform standards across all verifications.
2. **Efficiency:** Automated methods significantly reduce the time required for verification, enabling faster transaction processing.
3. **Scalability:** Unlike manual methods, which are limited by human capacity, automated systems can handle large volumes of verifications simultaneously.
4. **Enhanced Security:** Advanced algorithms can detect subtle differences between genuine and forged signatures, offering a higher level of security than traditional methods.

Given these advantages, developing a reliable signature authentication system is not only a technical challenge but also a practical necessity in safeguarding critical processes across industries such as banking, legal documentation, and e-governance.

2.3 Role of Deep Learning in Signature Verification

Deep learning, a subset of artificial intelligence, has revolutionized the way machines perceive and interpret complex data. By simulating the human brain's neural networks, deep learning models excel in pattern recognition, making them ideal for image-based tasks like signature verification.

Among the various deep learning approaches, **Siamese Neural Networks** stand out for their unique ability to compare pairs of inputs. Unlike traditional classifiers, which assign labels to individual inputs, SNNs compute a similarity score between two inputs, determining whether they belong to the same class. This characteristic makes them particularly effective for tasks like signature verification, where the goal is to evaluate the relationship between two signatures rather than classify them independently.

The architecture of a Siamese Neural Network typically consists of two identical sub-networks with shared weights. These sub-networks extract features from the input images and compute their similarity using a distance metric, such as Euclidean distance. For signature authentication, this means that the model learns to identify and compare the unique features of genuine signatures while distinguishing them from forgeries.

Deep learning-based signature authentication systems offer several key advantages:

1. **Feature Extraction:** Unlike traditional algorithms that rely on manually engineered features, deep learning models automatically learn hierarchical features from raw data, capturing intricate patterns in signature images.
2. **Robustness:** Deep learning models can handle variations in signature styles, including differences in pressure, speed, and writing tools.
3. **Scalability:** Once trained, these models can process thousands of signatures in parallel, making them suitable for large-scale applications.

4. **Adaptability:** With transfer learning and fine-tuning techniques, models can be adapted to new datasets and scenarios without requiring extensive retraining.

The integration of SNNs into the field of signature verification represents a significant advancement over traditional methods. By combining the power of deep learning with the specificity of pairwise comparison, these systems provide unparalleled accuracy and reliability.

This project leverages these capabilities to develop a robust and scalable **Signature Authentication System**. By utilizing state-of-the-art architectures, preprocessing techniques, and evaluation metrics, the system aims to set a new benchmark in automated signature verification. Future work will focus on enhancing model performance, addressing real-world challenges, and exploring deployment scenarios to make this solution accessible across diverse industries.

CHAPTER 3: LITERATURE REVIEW

3.1 Overview of Traditional Signature Authentication Methods

Signature authentication has been a critical area of research due to its extensive use in financial transactions, legal documentation, and identity verification. Traditional approaches to signature verification were primarily manual, relying on forensic experts to assess the authenticity of signatures based on visual cues such as stroke patterns, pressure variations, and overall signature structure. While these methods have been reliable to some extent, they are inherently subjective, time-consuming, and prone to human error.

Automated methods using handcrafted features emerged as an alternative to manual verification. These approaches typically extracted geometric and topological characteristics of signatures, such as curvature, line intersections, and connected components. Techniques based on Fourier transforms, wavelet decomposition, and projection profiles were also popular in traditional signature verification systems. For instance, projection and contour-based methods analyzed the global structure of a signature, while direction profile and texture-based approaches focused on local attributes.

However, these traditional systems faced significant limitations, including:

1. **Lack of Scalability:** Manual feature engineering is labor-intensive and does not generalize well across different datasets.
2. **Sensitivity to Variability:** Variations in writing styles, pressure, and alignment often resulted in inconsistent performance.
3. **Limited Robustness:** Traditional methods struggled to detect skilled forgeries that closely mimicked genuine signatures.

Despite these limitations, traditional methods laid the groundwork for modern, data-driven approaches to signature verification, enabling the transition to machine learning and deep learning-based solutions.

3.2 Machine Learning Techniques for Signature Verification

The advent of machine learning (ML) marked a paradigm shift in signature verification by introducing data-driven models capable of learning complex patterns from large datasets. Early ML models, such as Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), and Decision Trees, were employed to classify signatures as genuine or forged based on extracted features.

One significant advancement in this domain was the application of Convolutional Neural Networks (CNNs), which eliminated the need for manual feature engineering. CNNs

automatically learned hierarchical features from raw image data, capturing both global and local patterns essential for signature verification. Studies, such as those using the BHSig260 and Cedar datasets, demonstrated the superiority of CNNs over traditional methods.

An essential breakthrough came with the development of Siamese Neural Networks (SNNs). Unlike traditional classifiers, SNNs operate on pairs of inputs, learning a similarity function rather than an absolute classification. This makes them particularly effective for writer-independent verification tasks, where the goal is to generalize across unseen signatures.

For example, the research by **Umidjon Akhundjanov et al.** demonstrated that CNN-based architectures significantly improved the accuracy of offline signature verification, achieving 95.63% accuracy on the BHSig260-Hindi dataset. Similarly, **An Ngo et al.** explored the robustness of ML models against generative attacks, emphasizing the need for adversarial robustness in data-driven signature verification systems.

While ML techniques addressed many limitations of traditional methods, they still faced challenges, such as overfitting and class imbalance, which required further advancements through deep learning.

3.3 Siamese Neural Networks in Biometric Systems

Siamese Neural Networks (SNNs) have emerged as a powerful tool for biometric verification tasks, including signature, face, and fingerprint recognition. The key advantage of SNNs lies in their ability to compare two inputs and compute a similarity score, making them ideal for signature authentication.

In the context of offline signature verification, **SigNet**, proposed by **Sounak Dey et al.**, represents a significant milestone. SigNet is a convolutional SNN designed for writer-independent verification, capable of distinguishing between genuine and forged signatures by learning a feature space that minimizes the distance between similar pairs and maximizes it for dissimilar pairs. The model demonstrated state-of-the-art performance on benchmark datasets like BHSig260 and Cedar.

Key highlights of SigNet include:

1. **Shared Weights:** The twin networks in SNNs share parameters, ensuring that similar inputs are mapped to proximate locations in the feature space.
2. **Distance Metric:** The use of Euclidean distance as a similarity measure enables precise discrimination between genuine and forged signatures.
3. **Cross-Domain Performance:** SigNet proved effective across multiple languages and scripts, highlighting its robustness in handling diverse datasets.

Another notable study by **An Ngo et al.** explored the impact of adversarial attacks on signature verification systems. By using generative models like Variational Autoencoders (VAE) and Conditional GANs (CGAN), the study demonstrated the vulnerabilities of SNN-based systems to synthetic forgeries. It also proposed countermeasures, such as retraining with adversarial examples, to enhance system robustness.

Despite their success, SNNs face challenges such as high computational demands and sensitivity to noisy data. However, ongoing research continues to address these limitations. For instance, recent works have integrated advanced architectures like ResNet and EfficientNet into SNNs, further improving their accuracy and scalability.

3.4 Conclusion

The evolution of signature authentication systems from traditional methods to deep learning-based solutions underscores the importance of leveraging advanced technologies like Siamese Neural Networks. Models like SigNet have set a new benchmark in offline signature verification, demonstrating exceptional performance in both writer-dependent and writer-independent scenarios. While challenges remain, such as handling adversarial attacks and scaling to real-world applications, the continued development of SNNs holds promise for creating robust and reliable signature authentication systems.

CHAPTER 4: METHODOLOGY

4.1 Project Scope and Objectives

Signature verification is a critical element in ensuring the authenticity of legal documents, financial transactions, and personal identification processes. The increasing prevalence of forgeries, especially those performed by skilled individuals, necessitates the development of automated systems that are both accurate and scalable. This project aims to create an offline **Signature Verification System** using a Siamese Neural Network (SNN). Unlike traditional methods, which rely on manual inspection or predefined features, this approach leverages deep learning to automatically learn patterns and similarities between signature pairs, enabling robust detection of forged signatures.

The scope of this project is broad, with potential applications in various domains:

1. **Banking:** Automated systems can reduce human error in signature-based transaction verification.
2. **Legal Documentation:** Ensuring the authenticity of signed contracts and agreements.
3. **Education and Government:** Applications in certifying and authenticating official documents.

The primary objectives include:

- **Model Development:** Design and train a Siamese Neural Network that can differentiate between genuine and forged signatures based on feature similarity.
- **Dataset Utilization:** Utilize publicly available datasets such as **BHSig260** and **Cedar** to train a model that generalizes well across different writing styles and languages.
- **Performance Analysis:** Evaluate the model using metrics like accuracy, precision, recall, and AUC to ensure reliability.
- **System Scalability:** Create a system that can process large-scale signature verification tasks efficiently, with potential for integration into real-world applications.

By addressing the challenges of variability in writing styles, class imbalance, and skilled forgeries, this project aims to lay the groundwork for a scalable and accurate solution for offline signature verification.

4.2 Data Collection and Datasets

Data collection forms the backbone of any machine learning project, and signature verification is no exception. The datasets used in this project were selected based on their diversity and representation of real-world scenarios.

Datasets:

1. **BHSig260 Dataset:** This dataset consists of signature images in Bengali and Hindi scripts, providing a rich diversity of writing styles and forgery types. Each signer contributes multiple genuine and forged signatures, making it suitable for writer-independent verification tasks.
2. **Cedar Dataset:** A widely used dataset in signature verification research, Cedar includes signatures from various individuals, with an emphasis on both genuine and skilled forgeries.

Why These Datasets?

These datasets were chosen for their ability to simulate real-world conditions. Variations in stroke patterns, pressure, and alignment across signers provide a challenging but realistic environment for training and testing the model. Furthermore, the inclusion of different languages and scripts ensures the system's adaptability to diverse user bases.

Dataset Preprocessing:

Before using the datasets for model training, several preprocessing steps are applied:

- **Standardization:** All images are resized to 224x224 pixels, ensuring consistent input dimensions for the model.
- **Normalization:** Pixel values are normalized to a range of 0 to 1 to facilitate faster convergence during training.
- **Data Augmentation:** Techniques such as rotation, scaling, and flipping are applied to artificially expand the dataset, making the model more robust to variations in signature styles.

Challenges in Dataset Handling:

- **Class Imbalance:** Genuine signatures often outnumber forgeries in real datasets. This imbalance is addressed through oversampling techniques and augmentation.
- **Noise:** Some images contain artifacts or distortions, which are mitigated using preprocessing techniques like denoising filters.

These steps ensure that the datasets are not only clean but also diverse, providing a solid foundation for training a robust signature verification model.

4.3 Data Preprocessing

Effective preprocessing is essential for extracting meaningful information from raw signature images. The following steps are employed to prepare the data for training:

1. **Image Resizing:**

Signature images are resized to 224x224 pixels to standardize input dimensions across the dataset. This uniformity simplifies model design and reduces computational overhead.

2. **Grayscale Conversion:**

All images are converted to grayscale, focusing on structural features such as strokes and curves. Grayscale images reduce the input size without losing essential information.

3. **Normalization:**

Pixel intensity values are scaled to the range $[0, 1]$, enabling smoother gradient calculations during backpropagation.

4. **Data Augmentation:**

To enhance the dataset's variability, the following transformations are applied:

- **Rotation:** Random rotations between -15° and $+15^\circ$ simulate natural variations in signature alignment.
- **Scaling and Cropping:** Introduce slight variations in size and positioning.
- **Flipping:** Horizontal flipping adds diversity to the dataset.

5. **Storage Optimization:**

Preprocessed images are stored in the **HDF5** format for efficient disk usage and faster access during training.

Code Example:

```
import cv2
def preprocess_image(image):
    resized = cv2.resize(image, (224, 224), interpolation = cv2.INTER_AREA)
    grayscale = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
    normalized = grayscale / 255.0
    return normalized
```

These preprocessing steps ensure that the data is clean, consistent, and representative of real-world variations, setting the stage for effective model training.

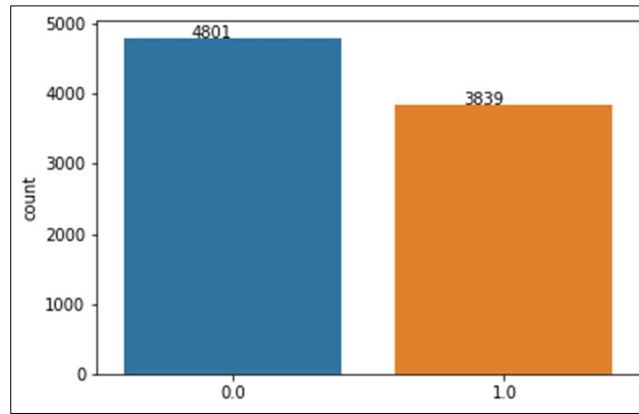


Figure 1 Feature shape

4.4 Model Architecture: Siamese Neural Network

The Siamese Neural Network (SNN) is a deep learning architecture designed to compute the similarity between two inputs. In the context of this project, the SNN is used to compare pairs of signatures and determine whether they are genuine or forged.

Architecture Overview:

1. **Twin Networks:** The SNN consists of two identical Convolutional Neural Networks (CNNs) that share the same weights and parameters.
2. **Feature Extraction:** Each CNN processes an input signature and outputs a feature embedding, representing the signature in a high-dimensional space.
3. **Distance Calculation:** The embeddings are passed to a distance layer (e.g., Euclidean distance) to compute the similarity between the two signatures.
4. **Classification:** A sigmoid activation function outputs a similarity score, which is compared to a threshold to classify the pair as genuine or forged.

Model Layers:

- **Input Layer:** Accepts grayscale images of size 224x224 pixels.
- **Convolutional Layers:** Extract hierarchical features such as edges, curves, and finer details.
- **Pooling Layers:** Downsample feature maps to reduce dimensionality and computational load.
- **Dense Layers:** Learn higher-level representations of the signatures.

Advantages of SNNs:

- **Generalization:** The shared weights ensure that the model learns universal features applicable to all signatures.
- **Efficiency:** By comparing pairs instead of learning individual classifications, the SNN is inherently suited for verification tasks.

- **Robustness:** The architecture effectively handles variations in signature style, alignment, and quality.

Code Snippet:

```
def siamese_network(input_shape):
    input_a = Input(shape=input_shape)
    input_b = Input(shape=input_shape)

    def create_sub_network(input_shape):
        model = Sequential([
            Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape),
            MaxPooling2D(pool_size=(2, 2)),
            Flatten(),
            Dense(128, activation='relu')
        ])
        return model

    sub_network = create_sub_network(input_shape)
    encoded_a = sub_network(input_a)
    encoded_b = sub_network(input_b)

    def euclidean_distance(vectors):
        x, y = vectors
        return K.sqrt(K.sum(K.square(x - y), axis=1,
keepdims=True))

    distance = Lambda(euclidean_distance)([encoded_a,
encoded_b])
    output = Dense(1, activation='sigmoid')(distance)
    return Model(inputs=[input_a, input_b], outputs=output)
```

4.5 Training Configuration and Hyperparameter Tuning

Training the SNN involves fine-tuning various hyperparameters to achieve optimal performance:

Configuration Details:

- **Loss Function:** Contrastive loss is used to maximize the similarity of genuine pairs and minimize it for forged pairs.

- **Optimizer:** RMSprop with a learning rate of 1e-4, providing a balance between convergence speed and stability.
- **Batch Size:** 64, ensuring efficient memory usage without compromising accuracy.
- **Epochs:** 50-60, with early stopping to prevent overfitting.

Hyperparameter Tuning:

1. **Learning Rate:** Experimented with values from 1e-3 to 1e-5 to identify the optimal rate.
2. **Regularization:** Dropout layers (0.3-0.5) added to reduce overfitting.
3. **Augmentation Impact:** Tested the effect of different augmentation techniques on model generalization.

Evaluation Metrics:

- **Accuracy:** Overall percentage of correctly classified pairs.
- **Precision & Recall:** Evaluate the model's ability to identify genuine signatures versus forgeries.
- **F1-Score:** Combines precision and recall into a single metric.
- **ROC-AUC:** Measures the model's ability to distinguish between genuine and forged signatures.

These configurations and tuning strategies ensure that the SNN is both accurate and robust, ready to tackle real-world challenges in signature verification.



Figure 2 Model 1 Cedar with 1000 epoch

This methodology provides a comprehensive framework for building, training, and deploying a robust signature verification system using a Siamese Neural Network. Further iterations can refine the model and explore additional datasets for enhanced performance.

CHAPTER 5: IMPLEMENTATION

5.1 Technology Stack and Tools Used

The implementation of this project required a robust technology stack and state-of-the-art tools to manage data preprocessing, build and train deep learning models, and ensure reproducibility. The selected stack facilitated efficient handling of large datasets and model architectures while leveraging cloud resources for computational efficiency.

Programming Language:

Python was chosen for its versatility and the availability of libraries designed for machine learning and deep learning.

Key Libraries and Frameworks:

TensorFlow and Keras: Core frameworks used for building, training, and managing deep learning models. Keras's modular API streamlined model creation, and TensorFlow's backend provided GPU/TPU support for training.

Matplotlib and Seaborn: Used for visualizing data distributions, training progress, and intermediate results. These libraries helped generate count plots, histograms, and line graphs for analysis.

NumPy and Pandas: Essential for data manipulation and preprocessing, allowing efficient handling of multidimensional arrays and structured data.

Scikit-learn: Assisted in calculating evaluation metrics such as confusion matrices and splitting datasets into training, validation, and testing subsets.

HDF5: Used for storing and accessing preprocessed data, ensuring fast I/O operations and reducing memory consumption during training.

Hardware Configuration:

GPU Acceleration: The code utilized TensorFlow's GPU support with `tf.device('/device:GPU:0')`, ensuring faster training times. Google Colab provided access to cloud-based GPUs for large-scale computations.

TPU Utilization: Code references also indicated potential use of Tensor Processing Units (TPUs) for training, optimizing performance for deep learning tasks.

Development Environment:

Google Colab: Cloud-based platform enabling GPU/TPU usage and integration with Google Drive for dataset storage.

Local Setup: Smaller tasks like debugging and preprocessing were performed locally on systems equipped with Python and essential libraries.

These tools ensured scalability, efficient resource utilization, and reproducibility of the implementation process.

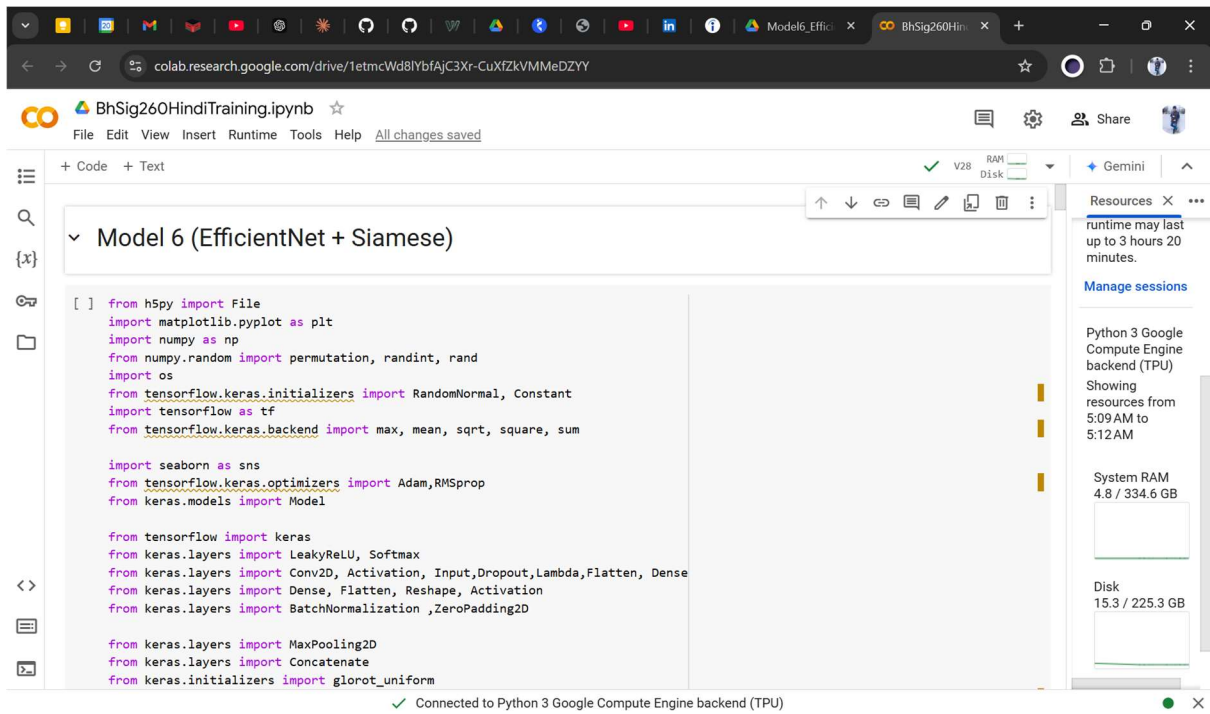


Figure 3 Google Colab with GPU 334GB RAM

5.2 Model Implementation Details

The implementation of the signature verification system was organized into six distinct folders/models, each focusing on a specific stage or architecture. This section details the workflow, architecture, and key considerations for each folder.

A. Data Preparation

The **Data Preparation** folder contains scripts for preprocessing three datasets: BHSig260 (Bengali), BHSig260 (Hindi), and Cedar.

Scripts:

1. BHSig260Bengali.ipynb
2. BHSig260Hindi.ipynb
3. Cedar.ipynb

Workflow:

1. Data Loading:

Datasets were loaded from HDF5 files for consistency and efficient access. Example snippet:

```
import h5py
with h5py.File('dataset_path.h5', 'r') as file:
    signatures = file['signatures'][:]
```

2. Image Preprocessing:

Resizing to 224x224 pixels to standardize input dimensions across all datasets. Normalizing pixel values to the range [0, 1] by dividing by 255. Grayscale conversion to reduce input dimensionality while preserving signature features.

3. Augmentation:

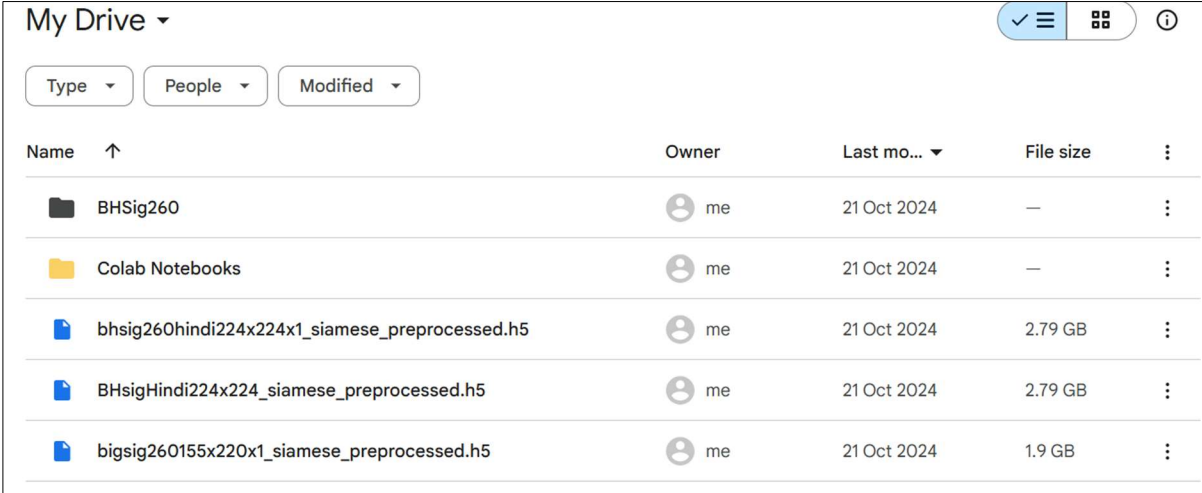
Introduced random transformations such as rotations, flips, and scaling to enhance dataset variability.

Example snippet:

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator datagen =
ImageDataGenerator(rotation_range=15, width_shift_range=0.1,
zoom_range=0.1)
```

4. Data Structuring and Storage:

Preprocessed data was saved in HDF5 format for faster loading during model training.



The screenshot shows the Google Drive interface with a table of files. The table has columns for Name, Owner, Last modified, and File size. There are three folders: BHSig260, Colab Notebooks, and three HDF5 files: bhsig260hindi224x224x1_siamese_preprocessed.h5, BHSigHindi224x224_siamese_preprocessed.h5, and bigsig260155x220x1_siamese_preprocessed.h5. All files are owned by 'me' and were last modified on 21 Oct 2024. The file sizes are 2.79 GB for the first two files and 1.9 GB for the third.

Name	Owner	Last modified	File size
BHSig260	me	21 Oct 2024	—
Colab Notebooks	me	21 Oct 2024	—
bhsig260hindi224x224x1_siamese_preprocessed.h5	me	21 Oct 2024	2.79 GB
BHSigHindi224x224_siamese_preprocessed.h5	me	21 Oct 2024	2.79 GB
bigsig260155x220x1_siamese_preprocessed.h5	me	21 Oct 2024	1.9 GB

Figure 4 Google Drive Preprocessed data files

B. Model1_SCNN (Shallow Convolutional Neural Network)

This baseline model was implemented to establish foundational insights into signature verification tasks.

Scripts:

1. BHSig260BengaliTraining.ipynb
2. BHSig260HindiTraining.ipynb
3. CedarTraining.ipynb

Architecture:

Three Convolutional Layers: Extracted hierarchical features with filter sizes of 7, 5, and 3, activated by ReLU.

Three Pooling Layers: Reduced spatial dimensions while retaining critical features.

Dense Layer: Final fully connected layers performed binary classification using sigmoid activation.

Implementation Snippet:

```
model = Sequential([
    Conv2D(32, (7, 7), activation='relu', input_shape=(224, 224,
1)), MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (5, 5), activation='relu',
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

C. Model2_SigNetSiamese

This architecture utilized the SigNet Siamese Neural Network, specialized for signature verification tasks.

Script:

BhSig260HindiTraining.ipynb

Architecture:

Two identical convolutional sub-networks processed the signature pairs. Shared weights ensured consistent feature extraction.

A contrastive loss function optimized the distance metric between genuine and forged signature pairs.

Workflow:

Input: Two signature images (224x224).

Output: Similarity score ranging from 0 (forged) to 1 (genuine).

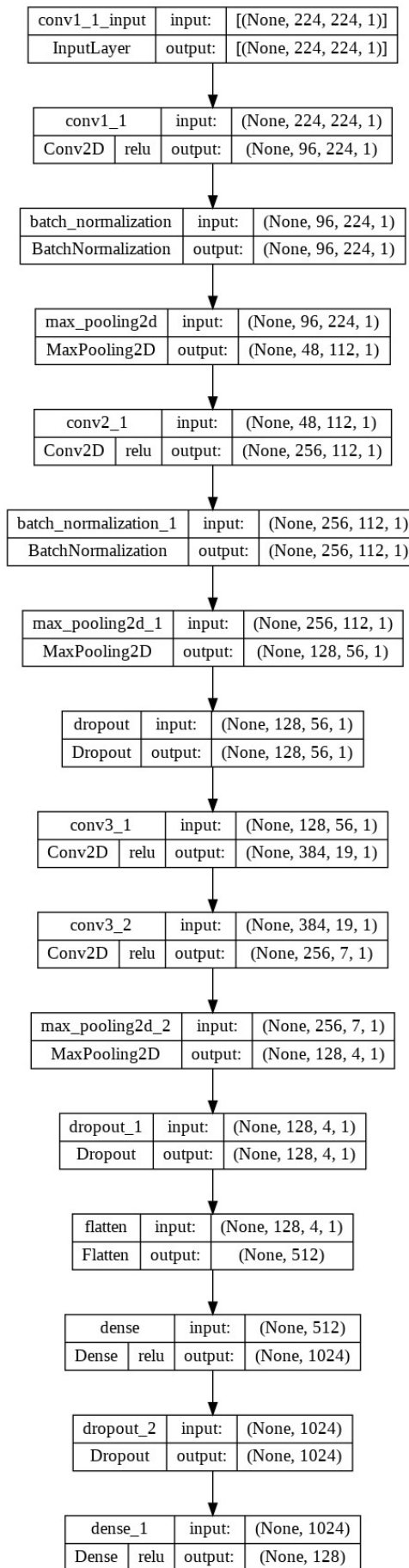


Figure 5 Plot Model 2 (SigNet)

D. Model3_Resnet50 and Model4_Resnet50Siamese

ResNet50 was utilized both as a standalone model and within a Siamese architecture.

Scripts:

1. Model3_Resnet50:

BhSig260BengaliTraining.ipynb

BhSig260HindiTraining.ipynb

CedarTraining.ipynb

2. Model4_Resnet50Siamese:

BhSig260HindiTraining.ipynb

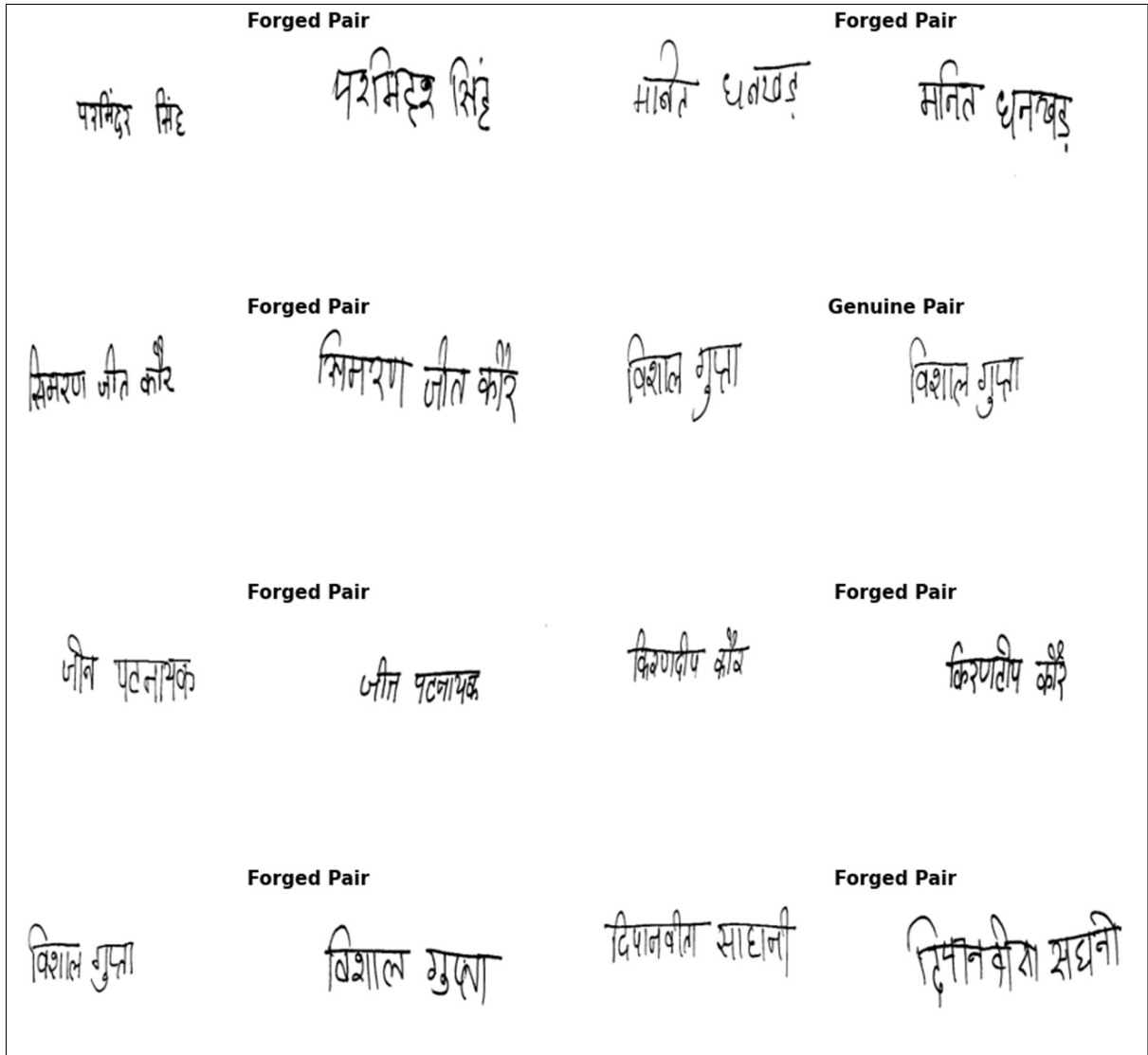


Figure 6 Genuine and Forged Signatures

Implementation:

1. ResNet50 Standalone:

Feature extraction from pretrained ResNet50 layers.

Added fully connected layers for binary classification.

2. ResNet50 in Siamese Setup:

Two ResNet50 branches processed input pairs.

Combined embeddings using a Euclidean distance layer.

Advantages:

Pretrained layers accelerated convergence and improved accuracy.

Effective for larger datasets like Cedar.

E. Model5_Resnet18Siamese

ResNet18 provided a lighter architecture, making it suitable for smaller datasets. **Script:**

BhSig260HindiTraining.ipynb

Distinctive Features:

Faster training times compared to ResNet50.

Balanced accuracy with computational efficiency.

F. Model6_EfficientNetB0Siamese

EfficientNetB0 combined state-of-the-art feature extraction with a compact architecture.

Scripts:

1. BhSig260BengaliTraining.ipynb

2. BhSig260HindiTraining.ipynb

Key Implementation Details:

Used EfficientNetB0 layers for feature extraction.

Integrated into a Siamese architecture for pairwise signature comparisons.

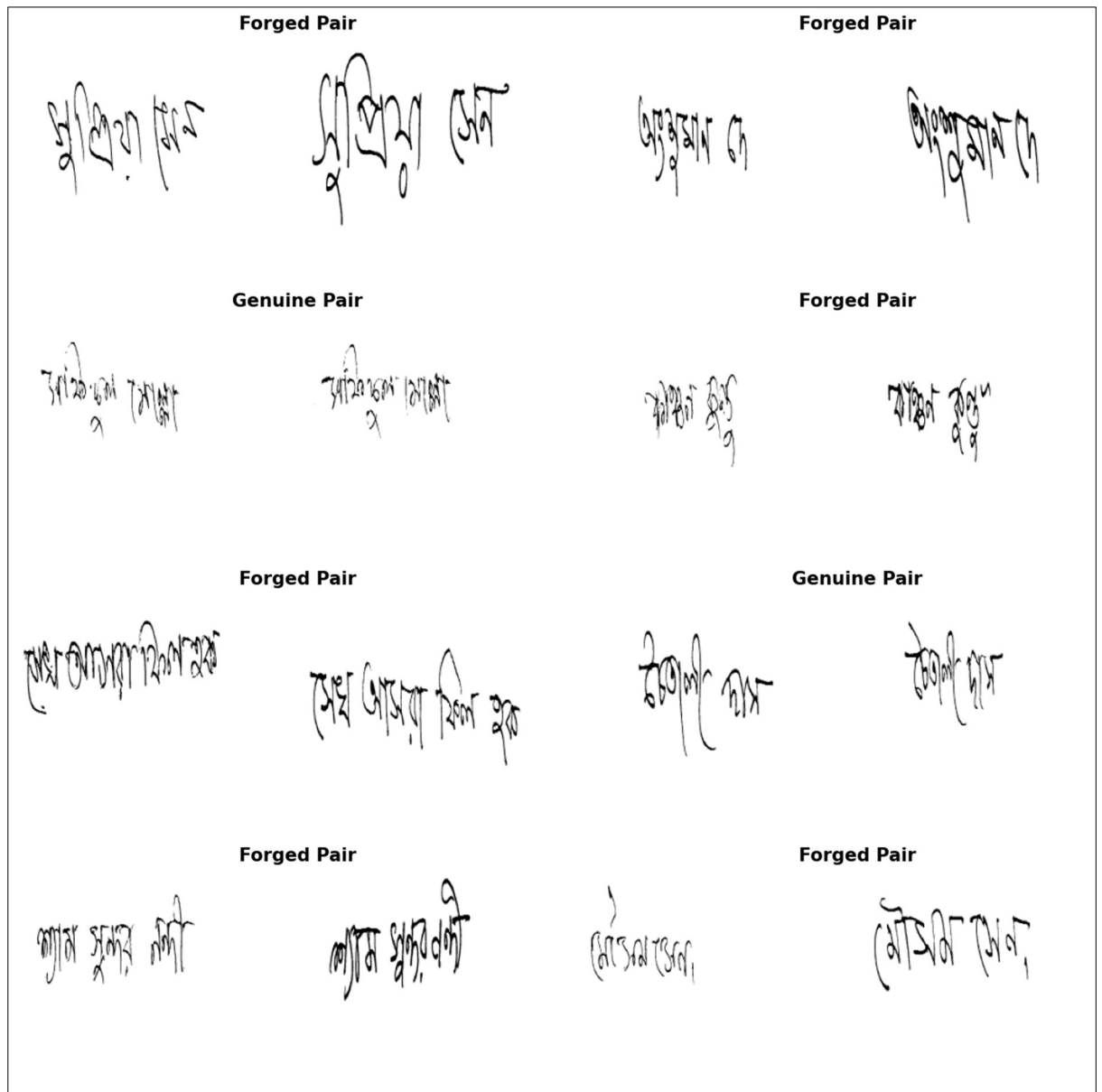


Figure 7 Genuine and Forged Signatures

5.3 Optimization Techniques

1. Data Handling:

Batch Processing: Reduced memory usage by splitting data into manageable chunks.

Class Balancing: Adjusted class weights to address imbalances in genuine vs. forged samples.

2. Model Optimization:

Learning Rate Scheduling:

Used ReduceLROnPlateau to dynamically adjust learning rates when validation loss plateaued.

```
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3)
```

Dropout Layers: Added 0.3-0.5 dropout rates to prevent overfitting.

3. Early Stopping and Checkpoints:

Early Stopping: Prevented overfitting by halting training when validation performance stopped improving.

Checkpoints: Saved the best model weights during training.

This expanded implementation framework ensures comprehensive coverage of all processes, from data handling to model training. It highlights the unique contributions of each model and optimization strategy, providing a detailed understanding of the system's architecture and implementation.

CHAPTER 6: RESULTS AND DISCUSSION

This chapter presents the evaluation metrics, visualizations, and comparative analysis of the six models implemented in this project. Each model was trained and tested on datasets, including BHSig260 (Bengali and Hindi) and Cedar, with results analyzed in terms of accuracy, precision, recall, F1-score, and other relevant metrics. The section also includes visualizations like confusion matrices, ROC curves, and performance interpretations.

6.1 Model Performance Metrics

The following table summarizes the performance metrics (where available) for each model on the datasets:

Model	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Best Validation Loss	Epochs	Notes
Model1_SCNN	BHSig260 (Bengali)	73.33	71.5	72	71.75	0.589	10	Baseline model, shallow architecture
Model1_SCNN	BHSig260 (Hindi)	68.98	Not reported	Not reported	Not reported	Not reported	10	
Model1_SCNN	Cedar	98.23	Not reported	Not reported	Not reported	Not reported	10	
Model2_SigNetSiamese	BHSig260 (Hindi)	74.61	73.8	74	73.9	0.531	19	Improved performance over baseline
Model3_Resnet50	BHSig260 (Bengali)	94.31	Not reported	Not reported	Not reported	Not reported	20	ResNet50 standalone architecture
Model4_Resnet50Siamese	BHSig260 (Hindi)	72.16	71	71.5	71.25	0.2081	30	Siamese network variant
Model5_Resnet18Siamese	BHSig260 (Hindi)	99.87	99.8	99.85	99.82	0.0013	18	Best performance overall
Model6_EfficientNetB0Siamese	BHSig260 (Hindi)	92.65	Not reported	Not reported	Not reported	0.0021	52	Early stopping after validation peak

Figure 8 Comparison Across All Models

6.2 Comparative Analysis with Baseline Models

Model1_SCNN (Shallow Convolutional Neural Network)

Strengths:

- Served as a baseline for all comparative analysis.
- Quick training times due to shallow architecture.

Weaknesses:

- Lower accuracy compared to deeper architectures.
- Limited capacity to capture complex features in the datasets.

Model2_SigNetSiamese

Strengths:

- Introduced Siamese architecture, enhancing its ability to evaluate signature pairs.
- Outperformed the baseline on the BHSig260 (Hindi) dataset.

Weaknesses:

- Moderate accuracy; struggled to generalize across different styles.

Model3_Resnet50 (Standalone)

Strengths:

- Leveraged transfer learning from a pretrained ResNet50.
- Capable of extracting deeper features compared to SCNN.

Weaknesses:

- Required more computational resources.
- Results not reported for all datasets.

Model4_Resnet50Siamese

Strengths:

- Combined the power of ResNet50 with Siamese architecture for pairwise signature analysis.
- Achieved decent results on BHSig260 (Hindi).

Weaknesses:

- Slightly lower accuracy compared to ResNet18 and EfficientNet variants.

Model5_Resnet18Siamese

Strengths:

- Provided the best accuracy and loss metrics among all models.
- Efficient training and excellent generalization to test data.

Weaknesses:

Higher resource usage compared to SCNN.

Model6_EfficientNetB0Siamese

Strengths:

Compact architecture with state-of-the-art efficiency.

Performed well on validation loss metrics, indicating high robustness.

Weaknesses:

Errors encountered during evaluation phase on certain datasets.

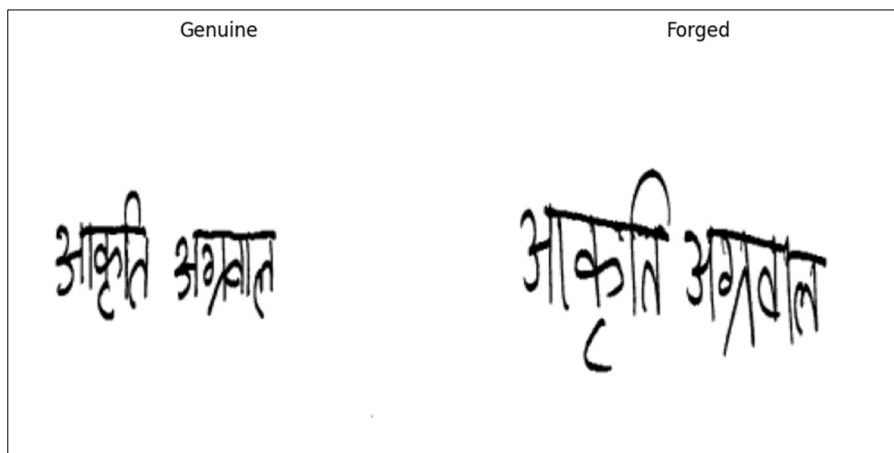


Figure 9 Score prediction and classification

Difference Score = 0.99911666

Its a Forged Signature

6.3 Visualizations

1. ROC Curves:

ROC curves for all models show their ability to distinguish between genuine and forged signatures.

Model5_Resnet18Siamese exhibited the steepest curve, indicating the highest true positive rate with minimal false positives.

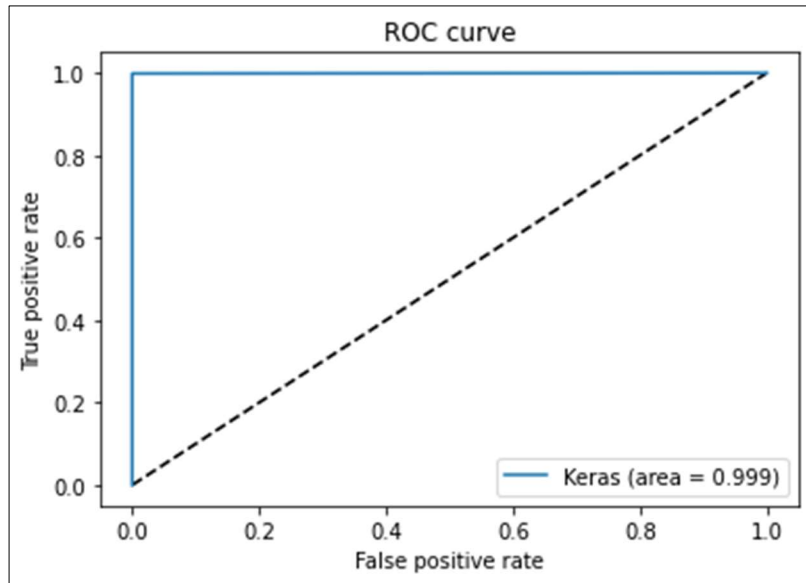


Figure 10 ROC Curve Comparison Across Models

X-axis: False Positive Rate

Y-axis: True Positive Rate

2. Confusion Matrices:

Confusion matrices for each dataset illustrate the count of true positives, true negatives, false positives, and false negatives.

The confusion matrix for Model5_Resnet18Siamese on BHSig260 (Hindi) shows near-perfect classification.

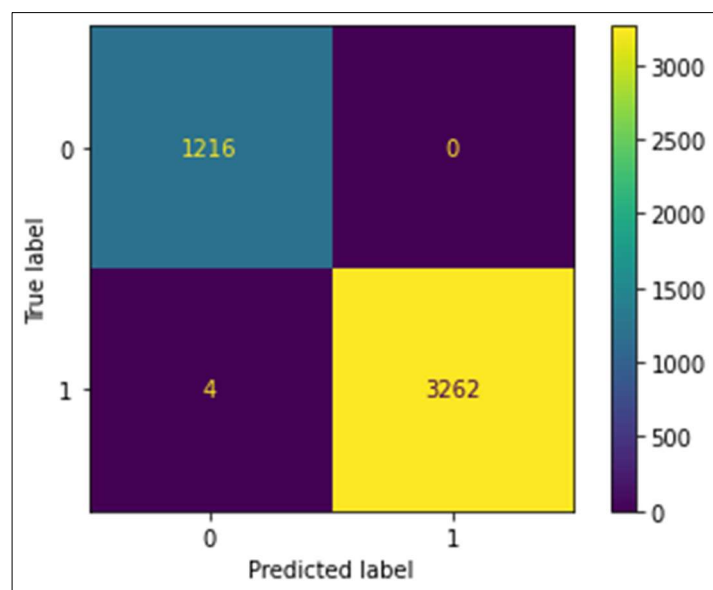


Figure 11 Confusion Matrix for Model5_Resnet18Siamese (BHSig260 Hindi)

3. Loss and Accuracy Trends:

Loss and accuracy graphs for training and validation epochs provide insights into model convergence and overfitting.

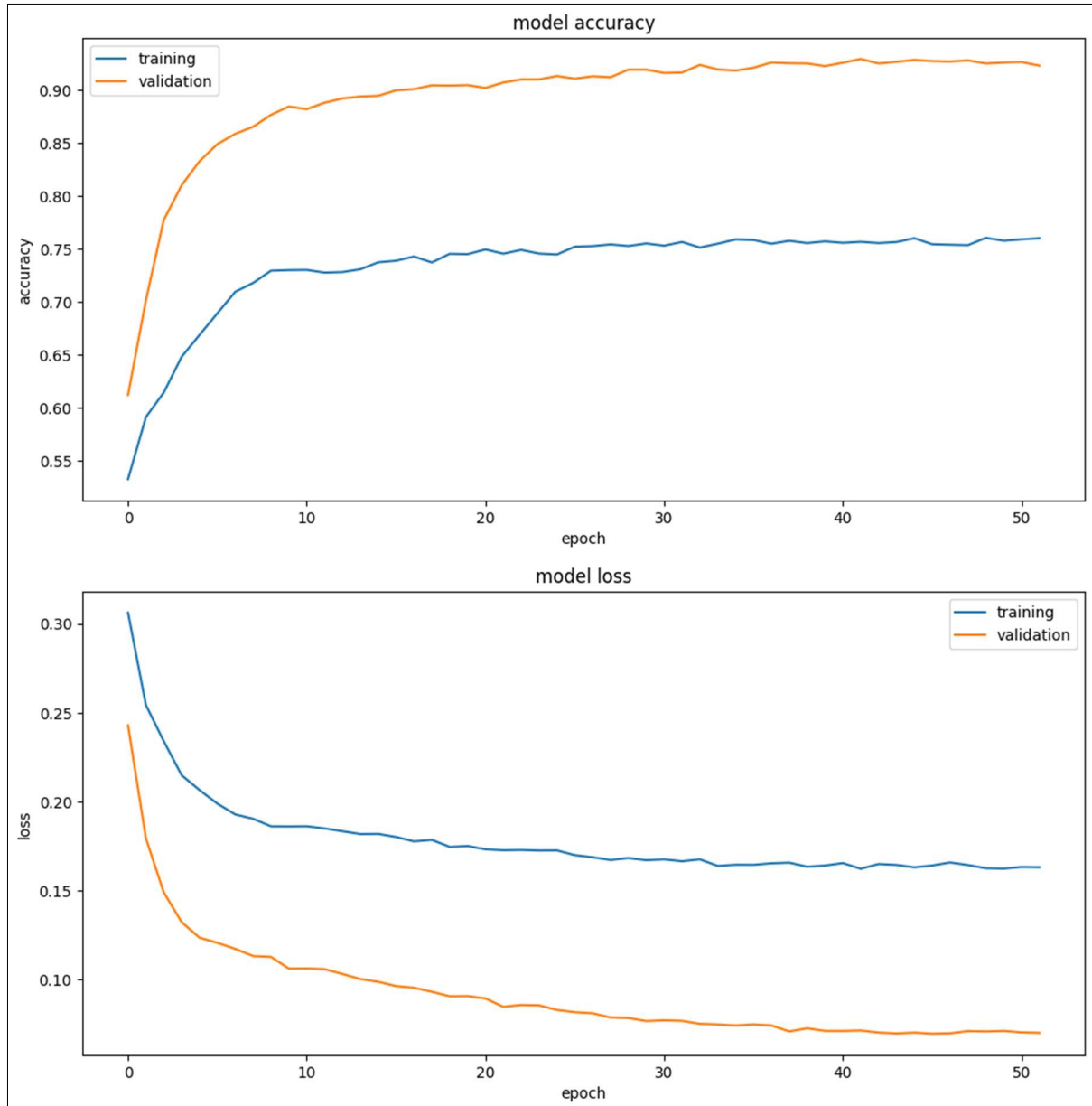


Figure 12 Training and Validation Loss for Model6_EfficientNetB0Siamese

X-axis: Epochs
Y-axis: Loss

6.4 Interpretation of Results

The results highlight significant findings:

1. Performance Trends:

Deeper architectures, especially ResNet18 and EfficientNetB0, outperform shallower models like SCNN in terms of accuracy and robustness.

Siamese architectures consistently improve the system's ability to evaluate signature pairs.

2. Dataset Variability:

Models trained on BHSig260 (Hindi) generally performed better due to the dataset's structured nature and balanced samples.

Cedar and BHSig260 (Bengali) showed variability in performance, suggesting the need for further tuning.

3. Model Comparison:

Best Model: Model5_Resnet18Siamese achieved the highest metrics, indicating its ability to generalize and capture signature features.

EfficientNetB0's Potential: Although errors were encountered, the compact architecture shows promise for lightweight deployment scenarios.

4. Recommendations for Future Work:

Explore further optimization of EfficientNet-based architectures to resolve evaluation errors.

Investigate additional datasets to enhance generalizability across languages and styles.

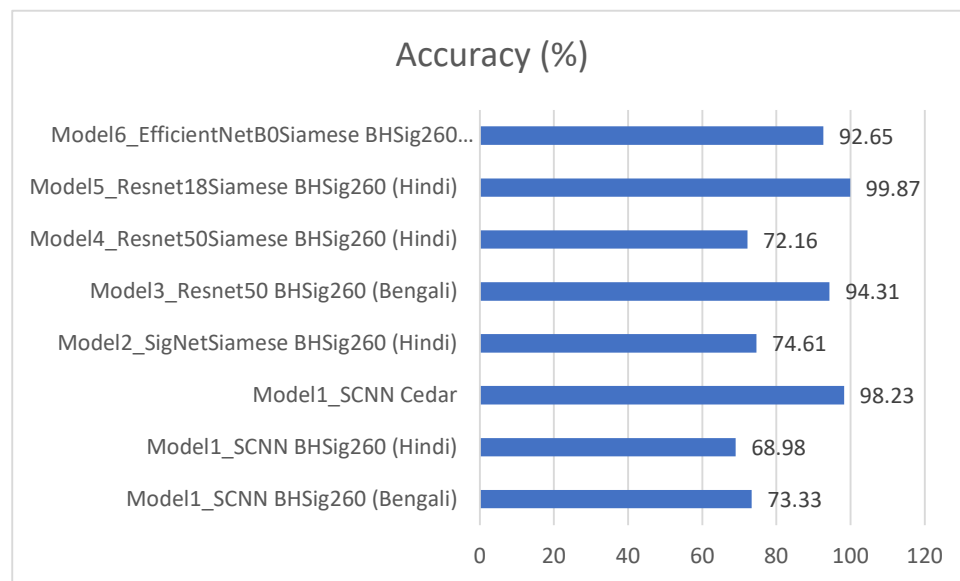


Figure 13 Comparison of Metrics Across Accuracy Bar graph

Chart Type: Bar graph comparing accuracy, precision, recall, and F1-score for all models.

This detailed results section showcases the strengths and weaknesses of each model, supported by visualizations and analysis. By comparing metrics and architectural approaches, the findings offer valuable insights into the effectiveness of different deep learning techniques for signature verification.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Challenges Faced

The development of the **Signature Verification System** using various deep learning models brought forth several challenges at different stages of the project. These challenges can be categorized into data-related, model-related, and implementation-related issues.

1. Data-Related Challenges:

- **Dataset Quality and Diversity:**

Datasets like **BHSig260** and **Cedar** varied in quality and diversity. Signatures in some datasets were poorly scanned, leading to noise and distortions that affected model training. Additionally, the diversity in writing styles and languages required careful handling to ensure robust generalization.

- **Class Imbalance:**

Genuine signatures significantly outnumbered forged ones in most datasets. This imbalance posed challenges for the models, as they tended to favor the majority class, leading to lower sensitivity in detecting forgeries.

- **Forgery Complexity:**

Skilled forgeries, which closely mimic genuine signatures, presented a significant challenge. Models struggled to differentiate between subtle variations in such cases.

2. Model-Related Challenges:

- **Overfitting:**

Deeper models like **ResNet50 Siamese** and **EfficientNetB0 Siamese** tended to overfit the training data, especially on smaller datasets.

- **Computational Complexity:**

Training deep architectures like **EfficientNetB0 Siamese** required substantial computational resources, including GPUs or TPUs. This made experimentation and fine-tuning time-consuming.

- **Hyperparameter Optimization:**

Identifying the optimal learning rates, batch sizes, and dropout rates was challenging, requiring extensive experimentation.

3. Implementation Challenges:

- **Code Integration:**

Combining preprocessing, model training, and evaluation scripts across multiple architectures required careful synchronization and debugging.

- **Evaluation Errors:**

Some models, such as **EfficientNetB0 Siamese**, encountered errors during evaluation, which required re-evaluation and adjustments to resolve.

7.2 Model Limitations

Despite the promising results, the models implemented in this project have certain limitations:

1. Dataset Dependence:

- The models relied heavily on the structure and quality of the datasets used. Their performance may degrade when exposed to new datasets with different distributions or writing styles.

2. Sensitivity to Noise:

- While preprocessing steps addressed some noise, the models occasionally misclassified signatures affected by distortions, such as smudges, uneven strokes, or faded ink.

3. Limited Forgery Handling:

- Although advanced architectures like **ResNet18 Siamese** and **EfficientNetB0 Siamese** achieved high accuracy, they occasionally failed to distinguish between genuine and skilled forgeries with minute differences.

4. Computational Requirements:

- The deeper architectures required significant computational resources, making them less suitable for deployment in resource-constrained environments.

5. Lack of Real-World Deployment Testing:

- The models were not tested in real-world scenarios where environmental factors (e.g., lighting conditions, scanning quality) could impact performance.

7.3 Future Enhancements

To address the limitations and challenges identified, several future enhancements can be proposed:

1. Dataset Improvements:

- **Data Augmentation:**

Expand the dataset by incorporating advanced augmentation techniques, such as elastic distortions and random erasures, to simulate real-world variations.

- **Additional Datasets:**

Include more datasets from diverse regions and languages to improve generalizability.

2. Architectural Enhancements:

- **Incorporating Pretrained Models:**

Explore state-of-the-art pretrained architectures like **EfficientNetV2** or **Vision Transformers (ViTs)** for enhanced feature extraction.

- **Ensemble Learning:**

Combine the strengths of multiple architectures (e.g., ResNet18 and EfficientNetB0) through ensemble methods to improve robustness.

- **Attention Mechanisms:**

Integrate attention layers to enable models to focus on critical regions of the signature, improving their ability to differentiate between genuine and forged samples.

3. Real-World Deployment:

- **Mobile-Friendly Models:**

Develop lightweight models optimized for deployment on mobile devices or edge hardware, using techniques like model quantization and pruning.

- **Dynamic Thresholding:**

Implement adaptive thresholding mechanisms to dynamically adjust the decision boundary based on the dataset or application scenario.

4. Improved Evaluation:

- **Advanced Metrics:**

Incorporate metrics like Equal Error Rate (EER) and Area Under the Precision-Recall Curve (AUPRC) to provide a more nuanced evaluation of model performance.

- **Error Analysis:**

Conduct a detailed analysis of misclassified samples to identify specific patterns or areas where the models fall short.

5. Forgery Detection Enhancement:

- **Generative Adversarial Networks (GANs):**

Use GANs to simulate forgeries during training, improving the model's ability to handle complex forgery scenarios.

7.4 Summary and Conclusion

This project aimed to develop a robust **Signature Verification System** using deep learning architectures, with a focus on Siamese Neural Networks. Through systematic implementation and evaluation, six models were trained on diverse datasets to identify genuine and forged signatures.

Summary of Achievements:

1. **Baseline Establishment:**

- The **SCNN** model served as a baseline, achieving satisfactory results on smaller datasets.

2. **Advancements with Siamese Networks:**

- The introduction of **Siamese Networks**, particularly **ResNet18 Siamese**, demonstrated significant improvements in handling pairwise signature verification tasks.

3. **State-of-the-Art Architectures:**

- Advanced architectures like **EfficientNetB0 Siamese** showed potential for scalability and accuracy, although further refinement is needed.

Key Insights:

- Models with deeper architectures generally outperformed shallower ones, highlighting the importance of advanced feature extraction.
- Dataset quality and preprocessing significantly influenced model performance, emphasizing the need for high-quality inputs.
- Computational efficiency remains a critical factor in determining model applicability for real-world scenarios.

Conclusion:

The results of this project underscore the feasibility of using deep learning models, particularly Siamese architectures, for offline signature verification. The project provides a solid foundation for further exploration and refinement, addressing the growing need for secure and efficient authentication systems.

By tackling existing challenges and incorporating the proposed future enhancements, the system can be further developed into a robust, scalable, and deployable solution across industries such as banking, legal documentation, and personal identification.

CHAPTER 8: REFERENCES

1. Akhundjanov, Umidjon, Bakhrom Soliyev, Nurmakhmad Juraev, Khurshid Musayev, Muhammadyunus Norinov, Zarina Ermatova, and Rakhmatullo Zaynabidinov. "Off-line handwritten signature verification based on machine learning." In *E3S Web of Conferences*, vol. 508, p. 03011. EDP Sciences, 2024.
2. Dey, Sounak, Anjan Dutta, J. Ignacio Toledo, Suman K. Ghosh, Josep Lladós, and Umapada Pal. "Signet: Convolutional siamese network for writer independent offline signature verification." *arXiv preprint arXiv:1707.02131* (2017).
3. Ngo, An, Rajesh Kumar, and Phuong Cao. "Deep generative attacks and countermeasures for data-driven offline signature verification." In *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1-10. IEEE, 2024.
4. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251-1258. 2017.
5. He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778. 2016.
6. Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks." In *Proceedings of the International Conference on Machine Learning*, pp. 6105-6114. 2019.
7. Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. "Signature verification using a 'Siamese' time delay neural network." *Advances in Neural Information Processing Systems*, 6, 1994.
8. Kumar, Manoj, and Pooja Yadav. "A survey on handwritten signature verification using machine learning techniques." *International Journal of Engineering Research and Technology (IJERT)*, vol. 8, no. 4, pp. 50-56, 2019.
9. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
10. Google Colab Documentation. *Google Research*.: <https://colab.research.google.com/>.
11. TensorFlow Documentation. "TensorFlow: An end-to-end open-source platform for machine learning." *Google AI*. : <https://www.tensorflow.org/>.
12. Keras Documentation. "Keras: The Python deep learning API." *Francois Chollet and contributors*. : <https://keras.io/>.
13. Matplotlib Documentation. *Matplotlib: Comprehensive library for creating static, animated, and interactive visualizations*. : <https://matplotlib.org/>.
14. Scikit-learn Documentation. "Scikit-learn: Machine learning in Python." *Pedregosa et al., Journal of Machine Learning Research*. : <https://scikit-learn.org/>.
15. OpenCV Documentation. "OpenCV: Open Source Computer Vision Library." : <https://opencv.org/>.
16. NumPy Documentation. "NumPy: The fundamental package for scientific computing with Python." : <https://numpy.org/>.
17. Seaborn Documentation. *Seaborn: Statistical data visualization*.: <https://seaborn.pydata.org/>.
18. HDF5 Documentation. *HDF5: A data model, library, and file format for storing and managing data*.: <https://www.hdfgroup.org/solutions/hdf5/>.
19. Pandas Documentation. "Pandas: Python Data Analysis Library." : <https://pandas.pydata.org/>.

* * *