

HANDWRITTEN PASSWORDS FOR TOUCHSCREEN BIOMETRICS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science Engineering

By

VISHAL REDDY VADDE (REG. NO. 40111441)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University By UGC

**Accredited with Grade "A++" by NAAC | 12 B Status
by UGC | Approved by AICTE**

JEPPIAR NAGAR, RAJIV GANDHISALAI,

CHENNAI – 600119

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **VISHAL REDDY VDDE (40111441)** who carried out the project entitled "**HANDWRITTEN PASSWORDS FOR TOUCHSCREEN BIOMETRICS**" under my supervision from NOV 2024 to APR 2024.



Internal Guide

Ms. P. YOGITHA, M.E.
HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Head of the Department, Sathyabama,
Chennai - 600 119.

Dr. L. Lakshmanan, M.E., Ph. D

Submitted for Viva voce Examination held on ¹³ APRIL 2024



Internal Examiner



External Examiner

DECLARATION

I **VISHAL REDDY VADDE** (reg no: 40111441) hereby declare that the Project report entitled "**HANDWRITING PASSWORDS FOR TOUCHSCREEN BIOMETRICS**" done by me under the guidance of Asst. Professor **Ms.R.YOGITHA** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 13/04/2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.R.Yogitha M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTARCT

In the realm of biometric authentication, this project presents a novel approach that integrates the unique characteristics of handwritten signatures with touchscreen technology to establish a secure and user-friendly password matching system. The system leverages the power of Convolutional Neural Network (CNN) transfer learning, employing architectures like ResNet and DenseNet121, to recognize and authenticate handwritten signatures. A Flask web application serves as the interface, enabling users to submit their handwritten signatures for verification. Successful matches grant access to the device, showcasing the potential of this innovative biometric approach. The project commences with data collection and preprocessing, where a diverse dataset of genuine handwritten signatures is gathered from users. The Flask web application acts as the user-facing component of the system. Users are prompted to submit their handwritten signatures image input. The submitted signature is subjected to preprocessing to ensure consistency and optimal input for the trained model. The trained model then evaluates the signature's authenticity and similarity to the user's genuine signature. Upon a positive match, the application displays a "Password Matched" message and simulates device unlock, further underscoring the usability and real-world applicability of the. Finally, this project pioneers an innovative biometric authentication system that merges the familiarity of handwritten signatures with touchscreen technology.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	TABLE OF CONTENTS	vi
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
3	REQUIREMENTS ANALYSIS	
	3.1 Hardware Requirements	7
	3.2 Software Requirements Specification Document	7
4	DESCRIPTION OF PROPOSED SYSTEM	
	4.1 Objective	10
	4.2 Proposed System	10
	4.3 Modules	10
	4.4 Architecture and Overall Design of Proposed System	11
	4.5 Performance Evaluation	15
5	IMPLEMENTATION	17
6	RESULTS, DISCUSSION & OUTPUT	19
7	CONCLUSION	21
	REFERENCES	22
	APPENDIX	24

Figure No.	Figure Name	Page No.
4.4.1	Architecture Diagram	11
4.4.2	ResNet50 Architecture	14
4.4.3	Example	14
5.1	Block Diagram	17
6.1	Performance Evaluation	19
6.2	Confusion Matrix	19
6.3	Flask Website	20
6.4	Output	20

CHAPTER 1

INTRODUCTION

In an era marked by technological advancements and the proliferation of digital interactions, the realm of security and authentication has evolved significantly. The convergence of touchscreen biometrics and transfer learning through Convolutional Neural Networks (CNNs) forms the foundation of this project's innovation. Touchscreens have become ubiquitous in modern devices, enabling natural and intuitive interactions. Integrating touchscreen-based signature input with biometric recognition promises a seamless user experience. Transfer learning, a technique that leverages pre-trained neural network models to address specific tasks, empowers the system to learn the intricate nuances of individual signatures efficiently.

A signature is a distinct form of a person's name, which mostly tries to identify itself as a unique form representing the person. But signatures can be identical, since many people can have the same name. Hence it tries to act as a unique identifier to authenticate that person's authenticity, usually in legal documents. It is a very important means of identification as it is used in many things such as legal documents, identification cards, cheque's, etc. It can be considered to be a legal mark which represents a particular person. Since a signature is so widely used, there are many malicious actors trying to forge the signature for some personal gain; hence a very good signature forgery detection methods are necessary. Forgery is a legal term which is used when the purpose of is to defraud anyone. A forged signature is a signature than has been determined to not be genuine. Nowadays, a handwritten signature is widely accepted as a form of confirmation form a person in any transactions or legal documents which require authentication from the person. Since a signature acts as a confirmation in any binding legal or financial process, its authenticity has to be established and it needs to be verified.

Signature forgery detection can be done mainly using online and offline methods. Online signatures are acquired using a touch screen device usually, and the trajectory and variations are recorded with time as the signature is being done and

extracts features from the signature and performs several checks by comparing with a predefined database containing the signature. Hence, online signature forgery detection usually gives a very high rate of successful signature forgery detections. Offline Signature verification is an authentication method that usually uses the dynamics of a person's handwritten signature to measure and analyze the physical activity of signing. It tries to detect forgeries by using feature extraction and checking for any discrepancies in the signature. It is less accurate than an online signature verification system.

Signature forgery is a growing concern in various domains such as finance, legal, and commercial settings. As signatures are commonly used for identity verification, document authentication, and approval of transactions, the falsification of signatures can lead to severe financial and legal consequences. Therefore, detecting forged signatures has become a crucial task in ensuring document security and preventing fraud.

In recent years, deep learning techniques have shown promising results in various image classification tasks. Convolutional neural networks (CNNs) have become particularly popular for image classification and have been applied in various applications, including signature verification. One of the widely used CNN architectures for signature verification is ResNet50.

In this project, we propose using ResNet50 for fake signature detection. We aim to build a model that can accurately identify whether a given signature is genuine or forged. We will use a dataset of signature images, preprocess them to ensure uniformity, and train the ResNet50 model using transfer learning. We will evaluate the performance of the model and compare it with other existing approaches to fake signature detection.

CHAPTER 2

LITERATURE REVIEW

a) **HANDWRITTEN SIGNATURES FORGERY DETECTION** Kshitij Swapnil Jain, et al. (2021) defined the objective of the paper to verify if the signature is original or forged and to understand the characteristics of the signatures and implement a system to detect if the signature is forged. Although signature forgery can be detected by highly skilled experts, high accuracy cannot be achieved due to the many variations in handwriting styles and professionalism of forgers. Automatic recognition system can be significantly more effective in verifying signatures with high accuracy and differentiate a genuine and forged signature. A CNN is used as a feature extractor and a classifier in the proposed method. The feature extractor extracts features from the input data via convolution filtering and down sampling. The authors assume that if a CNN is trained for classifying forged and genuine signatures, the trained CNN can extract effective features for distinguishing behavior characteristics of forgery, such as drawing the complicated sections of a signature with delays or hesitation. While deep networks can represent complex functions and learn features at different levels of abstraction, it will pose a problem when the gradient decreases exponentially and quickly reaches zero as the backpropagation continues from the final to the first layer. To overcome this, the authors use ResNet, which essentially skips certain steps or connections, which in turn allows the direct back propagation of the gradient and prevents the gradient from falling to a very small value quickly. The authors proposed method has successfully completed offline signature verification with increased efficiency and accuracy, as well as the ability to detect sophisticated forgeries. Signature forgery detection was done successfully using python and its libraries along with a solution based on Convolutional Neural Network (CNN) by the authors.

b) **Digital signature Forgery Detection using CNN** Kiran, Lakkoju Chandra, et al. (2021) aim to present an analysis of the methods for verifying the integrity of

visual media, i.e., the detection of manipulated images. Digital image forgery detection is commonly divided into active and passive techniques. The authors have studied active approaches to digital image forgery detection. Digital signature and watermarking are significant methods for digital image authenticity using active techniques. A signature consists of a combination of individual features and is the most significant means to verify the authenticity of the signature. It can work based on online or offline methods. Offline methods verify the identity of an individual by comparing the signature of a current query with reference signatures that have already been written. Online methods use dynamic knowledge related to the script and are more stable and have higher accuracy. Offline verification method verifies a signature after it has been produced. Online verification searches a database and compares it with previous samples to authenticate. The authors are implementing an online verification system and a dataset of 2000 images of forged and original images of signatures are divided and used using the RGB format. The RGB image is converted into a grayscale image and then into a binary image; then noise is removed and the image is resized to obtain the signature. After the preprocessing features are extracted, forming a .CSV file for each individual. A CNN is trained using the data with three input layers with different weights and biases and 3 hidden layers with a set of neurons and an output layer to show the final output (Genuine or Forged). They have also defined loss and optimizer to minimize the loss. And applied softmax to calculate the accuracy. And finally, after evaluating the model we will receive an output, as to whether the image is Genuine or Forged. The highest accuracy the authors obtained was 99.7%. The average accuracy is about 97.8%.

c) OFFLINE SIGNATURE FORGERY DETECTION USING CONVOLUTIONAL NEURAL NETWORK

Raj Balsekar, et al. (2020) proposed a system for signature recognition and verification based on CNN (Convolutional Neural Network). This paper has literature surveys on other papers for this system. The first author covers how signature verification requires help from both static and dynamic signatures. The second author covers the exploration of multiple explanations for offline signature verification. The third author covers how an individual signature can change multiple times over a period of time-based on behavioral changes. The

fourth author covers automatic signature verification systems based on current advancements in image processing and machine learning. The fifth author covers how to extract recent geometric parameters from preprocessed signatures for comparing among different people. This paper describes the proposed system that can distinguish whether a signature is real or fake. First, this system collects handwritten signatures to extract unique features for creating a knowledgebase for all the signatures. These handwritten signatures are from a dataset containing 750 signatures from 150 individuals providing 5 signatures each. Second, these signatures are stored in the form of a matrix, then converted from the RGB image to a black and white image with the help of a grayscale algorithm, and finally undergoing geometric transformation. Third, these pre-processed signatures features are extracted and compared to the signatures features stored in the system. Finally, these signature features are compared with the real signature to determine whether the signature is real or fake, and this process is done using CNN. In CNN, this signature undergoes multiple layers which are the convolutional layer, pooling layer and fully connected layer to get the results. We can conclude that this proposed system works very well.

d) Handwritten Signature Verification using Deep Learning Alajrami, Eman, et al. (2019) proposed a system for signature verification and forgery detection using deep learning. This paper discusses two types of signature verification, namely offline (or) static and online (or) dynamic verification. The related techniques and methods used by this system in the paper are template matching and Hidden-Markov model for signature verification. Whereas for forgery detection, we use CNN(Convolutional Neural Network). The proposed methodology in this paper based on behavioral changes in handwritten signatures and not psychological characteristics. In this methodology, at first signatures and their certain amount of unique characteristics are collected and used for creating a knowledge base for this signatures. Then, they undergo preprocessing where this signatures are resized and distributed in two subdirectories. Finally, these images are loaded into CNN, which uses Keras with the TensorFlow backend and performance is evaluated. This paper also discusses CNN operations which are convolution, max pooling, fullyconnected

layer, ReLu(Rectifier) and softmax. The dataset used for this methodology is 300 images, of which 150 were real and forged, and are from 30 people with real signatures. The result of this methodology is a dataset with the split ratio of 8:2 that has the highest accuracy from 99.7 to 99.9%. We can conclude that this methodology has been implemented successfully.

e) Offline Signature Recognition and Forgery Detection using Deep Learning

Poddar, Jivesh, et al. (2020) proposed a system for signature recognition, forgery detection and verification based on CNN(Convolutional Neural Network), Crest-Through Method, SURF algorithm and Harris corner detection algorithm. In this system, CNN and Crest-Through Method are used for recognition and verification. Whereas, SURF and Harris corner detection algorithms are used for forgery detection. This paper has literature surveys on other papers also. These signatures undergo pre-processing using CNN and Crest-Through Method. In CNN, the signatures go through noise removal, scaling, centralization and rotation. Whereas in CrestThrough Method, the signatures go through length to space ratio, width to space ratio and Crest-Through parameter. Both CNN and Crest-Through Method are also used for signature recognition. Then, this signatures go through forgery detection. In forgery detection, Harris corner detection algorithm is used to detect and extract corner points of the signature for comparison with real signature. Whereas, SURF algorithm is used to detect and extract index points of the signature for comparison with real signature. The result of this system is that it has 94% and 85-89% accuracy for signature recognition and forgery detection, respectively. We can conclude that this system may have some flaws, but it is a useful one.

CHAPTER 3

HARDWARE AND SOFTWARE REQUIREMENT

3.1 HARDWARE REQUIREMENT:

- Processor: Intel® Core™ i3-2350M CPU @ 2.30GHz Installed memory (RAM):4.00GB
- System Type: 64-bit Operating System
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

3.2 SOFTWARE REQUIREMENT:

Python:

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

PIP:

It is the package management system used to install and manage software packages written in Python.

NumPy:

NumPy is a general-purpose array-processing package. It provides a high performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains

various features including these important ones: A powerful N-dimensional array object

- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas:

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python. We can analyze data in pandas with

1. Series
2. Data frames

Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and macOS. Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

Tensor Flow: Tensor flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous

implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 OBJECTIVE

- Utilizing Convolutional Neural Networks (CNNs) and transfer learning, the project aims to train a model capable of accurately verifying handwritten signatures.
- The project seeks to seamlessly merge the familiarity of handwritten signatures with modern touchscreen biometrics

4.2 PROPOSED SYSTEM:

- ☐ **Data Collection and Preprocessing:** Describe the process of collecting a diverse dataset of genuine handwritten signatures from users.
- ☐ **Preprocessing:** Augment the dataset using various transformations to increase its size and diversity, thereby improving the model's ability to generalize to new samples. Implement techniques such as image rotation, translation, scaling, and shearing to simulate real-world variations in signatures.
- ☐ **CNN Transfer Learning training with Augmented Data:** Use the augmented dataset for transfer learning on the pre-trained CNN architecture (e.g., ResNet or DenseNet121).
- ☐ **Model Evaluation:** Accuracy and loss.
- ☐ **Flask Web Application:** Design the user interface of the Flask app for signature submission.
- ☐ **Device Unlock Simulation**

4.3 MODULES

- Data Collection
- Data Preprocessing
- Model Training
- Model Testing

- Flask Framework Prediction

4.4 ARCHITECTURE

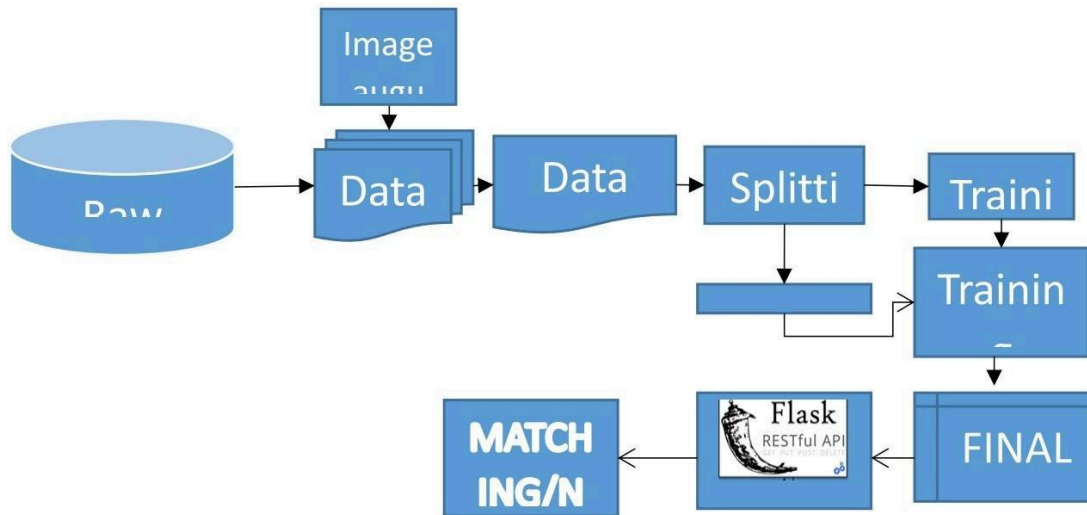


Fig.4.4.1: Architecture Diagram

MODULE-1:

DATA COLLECTION:

Contains Genuine and Forged signatures of 30 people. Each person has 5 Genuine signatures which they made themselves and 5 Forged signatures someone else made.

The naming of images is explained here. NFI-00602023 is an image of signature of person number 023 done by person 006. This is a forged signature. NFI-02103021 is an image of signature of person number 021 done by person 021. This is a genuine signature.

MODULE-2

DATA PREPROCESSING:

•Convolutional Neural Networks (CNs) can do wonders if there may be enough statistics. However, the choice of the right education information set of all of the characteristics which are essential for training is a hard undertaking. If the

consumer does now not have enough, the network can load schooling information. Realistic photographs contain distinct sizes, positions, zoom, lighting, noise, and so forth.

- For a robust community of these commonly encountered elements, the Data Augmentation method is used. By rotating input pictures to one-of-a-kind angles, flipping pictures via one of a kind axes, or rotating/cropping pix, the community will come across these phenomena in practice.

- Using Keras, there may be a handy institution of arguments in the "Image Data Generator" that allows picture augmentation. By flipping the photo or vertically, we can absolutely rearrange the factors, however the capability of the system is retained. That may be performed in numpy.

- Rotation is an increment this is typically accomplished at angles of 90 levels, but can also be done at smaller or minute angles if extra-large facts is needed. In fact, the history colour of the rotation is normally fixed so that it blends into the historical past. Otherwise, the model may also count on that the trade course is separate. This works excellent while the difficulty is the same in all turned around pix.

- Keras Image Data Generator class has two most important parameters width_shift_range and height_shift_range and based totally on those values the photo factors will shift both left or up and down. • At the time of schooling, it best includes studying one repressors or one binary classifier according to class. This being the case, it's far essential to label the multi-genres into binaries (they belong or do not belong to a genre). LabelBinarizer makes this method easy with the remodel technique.

- Convert to array and normalize to interval of [0,1]
- Fetch images and class labels from files
- Arrange format as per keras
- Resize as per model
- Append image
- Append class label

- Make labels into categories - either 0 or 1, for our model
- Label binarizer

MODULES -3

MODEL TRAINING

ResNet50 is a deep learning model that has achieved state-of-the-art performance in various computer vision tasks, including image classification, object detection, and segmentation. ResNet50 architecture uses residual connections, which allow the network to learn from both the original image and the residual image. This architecture enables the network to learn more complex features and significantly reduce the risk of vanishing gradients.

In the context of fake signature detection, ResNet50 can be used to identify the subtle differences between genuine and forged signatures. The model can automatically learn the features that are important for signature verification, such as the shape and structure of the signature strokes, which traditional machine learning-based approaches may not be able to capture accurately.

One of the advantages of using ResNet50 for fake signature detection is the ability to use transfer learning. Transfer learning is a technique where a pre-trained model on a large dataset, such as ImageNet, is used as a starting point for a new task. By using transfer learning, we can leverage the features learned by the pre-trained ResNet50 model and fine-tune them for signature verification, even with a relatively small dataset.

ResNet50 Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig.4.4.2: ResNet50 Architecture

Now we are going to discuss about Resnet 50 and also the architecture for the above talked 18 and 34 layer ResNet is also given residual mapping and not shown for simplicity.

There was a small change that was made for the ResNet 50 and above that before this the shortcut connections skipped two layers but now they skip three layers and also there was 1 * 1 convolution layers added that we are going to see in detail with the ResNet 50 Architecture.

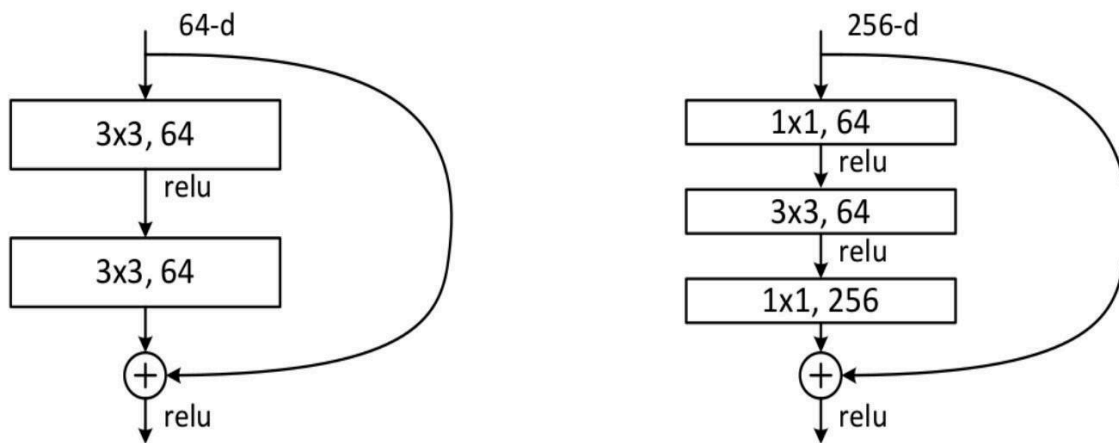


Fig.4.4.3: Example

So as we can see in the table 1 the resnet 50 architecture contains the following element:

- A convolution with a kernel size of $7 * 7$ and 64 different kernels all with a stride of size 2 giving us **1 layer**.
- Next we see max pooling with also a stride size of 2.
- In the next convolution there is a $1 * 1,64$ kernel following this a $3 * 3,64$ kernel and at last a $1 * 1,256$ kernel, These three layers are repeated in total 3 time so giving us **9 layers** in this step.
- Next we see kernel of $1 * 1,128$ after that a kernel of $3 * 3,128$ and at last a kernel of $1 * 1,512$ this step was repeated 4 time so giving us **12 layers** in this step.
- After that there is a kernel of $1 * 1,256$ and two more kernels with $3 * 3,256$ and $1 * 1,1024$ and this is repeated 6 time giving us a total of **18 layers**.
- And then again a $1 * 1,512$ kernel with two more of $3 * 3,512$ and $1 * 1,2048$ and this was repeated 3 times giving us a total of **9 layers**.
- After that we do a average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us **1 layer**.

We don't actually count the activation functions and the max/ average pooling layers.

so totaling this it gives us a $1 + 9 + 12 + 18 + 9 + 1 = 50$ layers Deep Convolutional network.

4.5 PERFORMANCE EVALUATION

MODEL TESTING:

Classification Accuracy: Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{\text{Number of Correct predictions}}{\text{Total number of predictions made}}$$

It works well only if there are equal number of samples belonging to each class.

Logarithmic Loss: Logarithmic Loss or Log Loss, works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

$$\text{LogarithmicLoss} = \frac{-1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} * \log(p_{ij})$$

where, y_{ij} , indicates whether sample i belongs to class j or not, p_{ij} , indicates the probability of sample i belonging to class j .

Log Loss has no upper bound and it exists on the range $[0, \infty)$. Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy. In general, minimising Log Loss gives greater accuracy for the classifier.

CHAPTER 5

IMPLEMENTATION

FLASK FRAMEWORK

The framework is the basis upon which software programs are built. It serves as a foundation for software developers, allowing them to create a variety of applications for certain platforms. It is a set of functions and predefined classes used to connect with the system software and handle inputs and outputs.

It simplifies the life of a developer while giving them the ability to use certain extensions and makes the online applications scalable and maintainable.

Frontend Development vs Backend Development

1. The front end of a website is the area with which the user immediately interacts. It contains everything that users see and interact with: text colours and styles, images and videos, graphs and tables, the navigation menu, buttons, and colours. HTML, CSS, and JavaScript are used in developing the front end.
2. The backend of a website refers to the server-side of the website. It saves and organizes data and ensures that everything on the client-side of the website functions properly. It is the section of the website that you are unable to interact with or access. It is the part of the software that has no direct communication with the users. Back End development is done using Java, Python, C#, and JavaScript.

Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Advantages of using Flask framework are:

- There is a built-in development server and a fast debugger provided.
- Lightweight
- Secure cookies are supported.
- Templating using Jinja2.
- Request dispatching using REST.
- Support for unit testing is built-in.

Flow diagram

Use Cases are used to describe the visible interactions that the system will have with users and external systems. They are used to describe how a user would perform their role using the system.

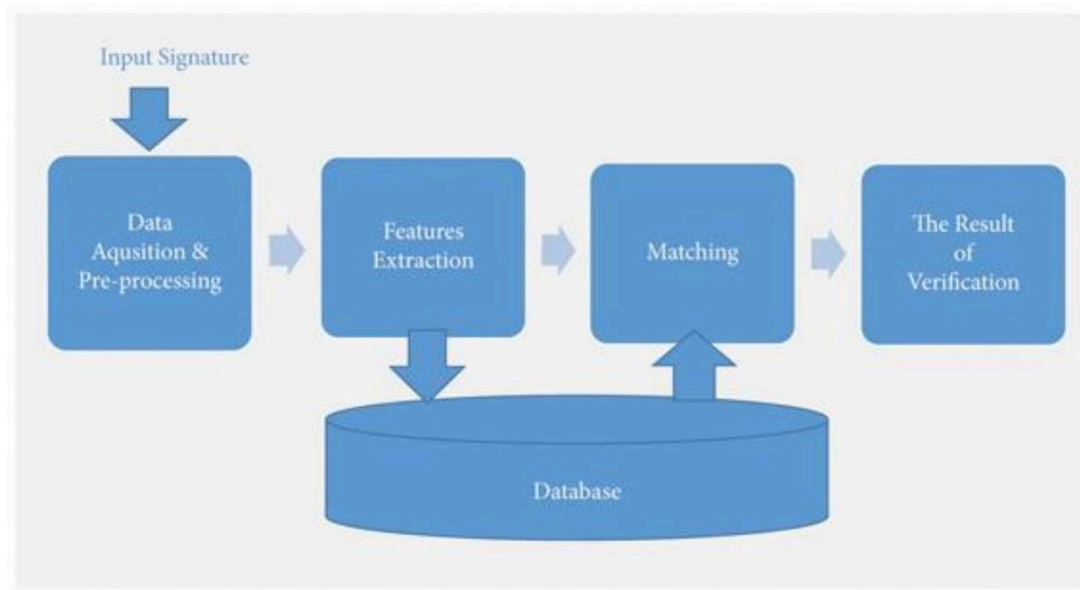


Fig. 5.1: Block Diagram

CHAPTER 6

RESULTS, DISCUSSION & OUTPUT

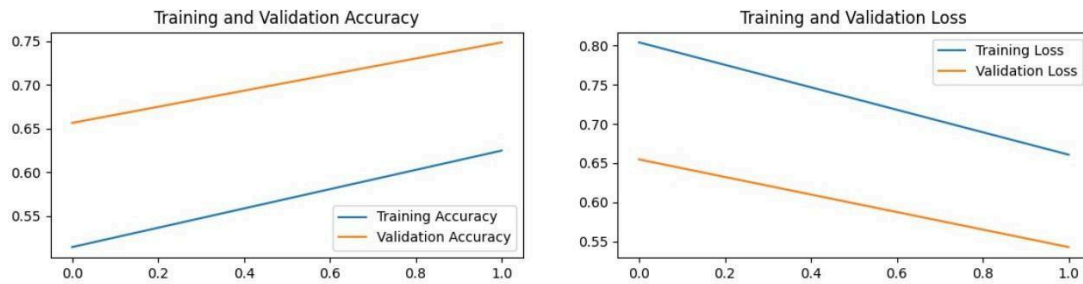


Fig. 6.1 Performance Evaluation

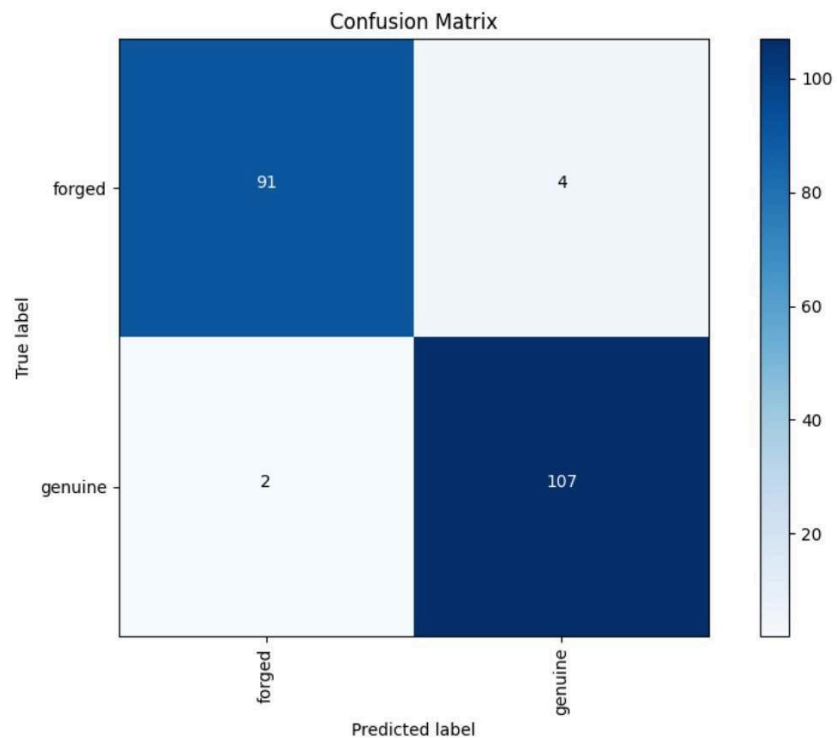


Fig. 6.2: Confusion Matrix

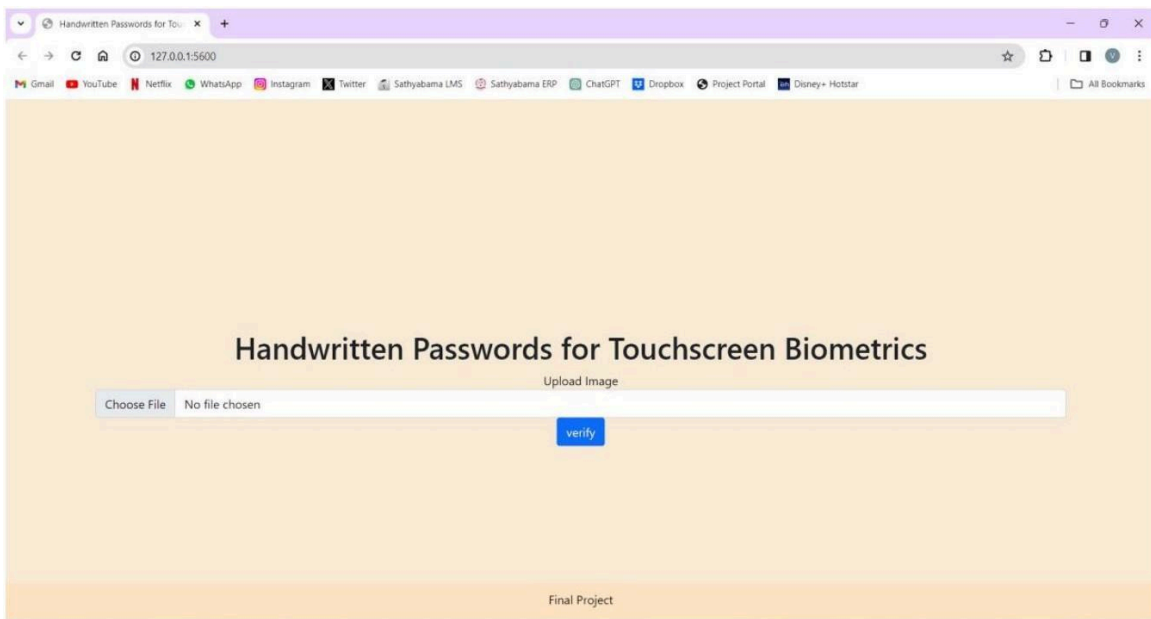


Fig. 6.3: Flask Website

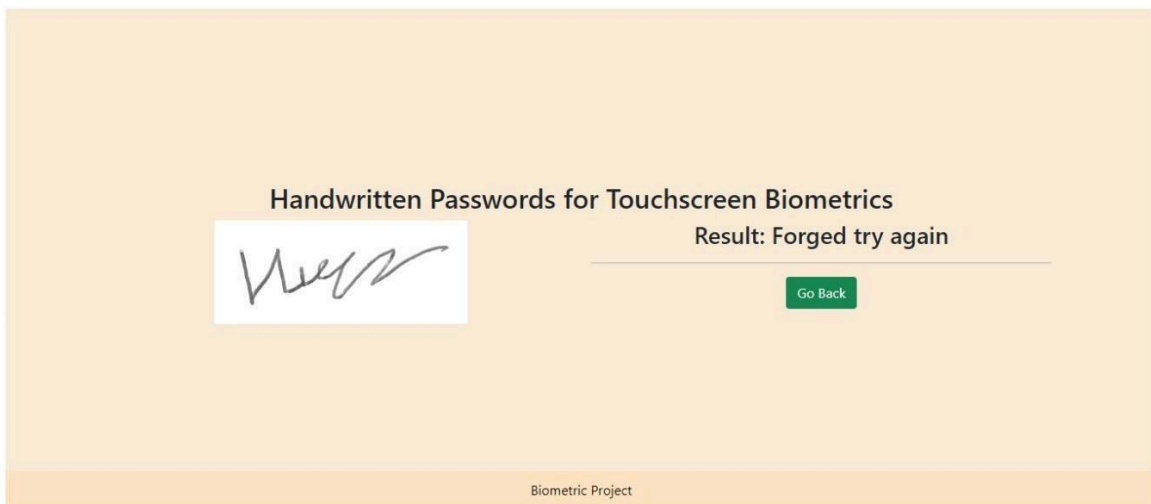


Fig. 6.4: Output

CHAPTER 7

CONCLUSION

Deep learning-based approaches, particularly CNNs, offer a potential solution to these limitations, and ResNet50, in particular, has shown great success in various computer vision tasks. Our project aimed to develop a password recognition in signature based detection system using ResNet50. Till now data is collected and preprocessing module completed.

A model that can learn from signatures and make predictions as to whether the signature in question is a forgery or not, has been successfully implemented. This model can be deployed at various government offices where handwritten signatures are used as a means of approval or authentication. While this method uses CNNs to learn the signatures, the structure of our fully connected layer is not optimal. This implementation may be considered extreme. In the model created in this work, two classes are created for each user (Real and forgery). We have 30 users and thus we have a model with 60 classes to predict. The best accuracy we got was 99.7%.

REFERENCES

- [1] Kshitij Swapnil Jain, et al. "HANDWRITTEN SIGNATURES FORGERY DETECTION" (2021). International Research Journal of Engineering and Technology (IRJET) Volume: 08 Issue: 01 | Jan 2021.
- [2] Kiran, Lakkoju Chandra, et al. "Digital signature Forgery Detection using CNN." (2021).
- [3] Raj Balsekar, et al. "OFFLINE SIGNATURE FORGERY DETECTION USING CONVOLUTIONAL NEURAL NETWORK", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.7, Issue 5, page no.13-18, May-2020.
- [4] Alajrami, Eman, et al. "Handwritten signature verification using deep learning." International Journal of Academic Multidisciplinary Research (IJAMR) 3.12 (2020).
- [5] Poddar, Jivesh, Vinanti Parikh, and Santosh Kumar Bharti. "Offline signature recognition and forgery detection using deep learning." Procedia Computer Science 170 (2020): 610-617.
- [6] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," Pattern Recognition, vol. 70, pp. 163–176, 2017.
- [7] M. Kumar, "Signature Verification Using Neural Network," International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 09, Sep 2012.
- [8] L. Batista, E. Granger, and R. Sabourin, "Dynamic selection of generative–discriminative ensembles for off-line signature verification," Pattern Recognition, vol. 45, no. 4, pp. 1326—1340, 2012.
- [9] H. Khalajzadeh, M. Mansouri, and M. Teshnehlal, "Persian signature verification using convolutional neural networks," International Journal of Engineering Research and Technology, vol. 1, no. 2, pp. 7–12, 2016.
- [10] M. B. A. Miah, M. A. Yousuf, M. S. Mia, M. P. Miya, "Handwritten Courtesy Amount and Signature Recognition on BankCheque using Neural Network," International Journal of Computer Applications, vol. 118, no. 5, 2015.

- [11] Y. Guerbai, Y. Chibani, and B. Hadjadji, "The effective use of the one-class svm classifier for handwritten signature verification based on writer-independent parameters," *Pattern Recognition*, vol. 48, no. 1, pp. 103–113, 2015.
- [12] B. Cozzens, R. Huang, M. Jay, K. Khembunjong, S. Paliskara, F. Zhan, M. Zhang, and S. Tayeb, "Signature Verification Using a Convolutional Neural Network," *BigData*, pp. 2644–2651, 2017.
- [13] M. Rezaei and N. Naderi, "Persian Signature Verification using Fully Convolutional Networks", 2019.
- [13] J. Darbon et al., "Fast nonlocal filtering applied to electron cryomicroscopy," in *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Proceedings, ISBI, IEEE*, pp. 1331–1334, 2008.
- [14] A. S. Bajestani, K. Fouladi, Kazim, and B. Araabi, "UTSig: A Persian offline signature dataset," Preprint, submitted to *IET biometrics*, 2016.
- [15] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization." University of Toronto, 2015.

APPENDIX

A. SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from PIL import Image
from tensorflow.keras.applications import Xception
# Tensorflow Libraries
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50

from keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam

from tensorflow.keras import Model
from tensorflow.keras.layers.experimental import preprocessing

# System libraries
from pathlib import Path
import os.path

# Metrics
from sklearn.metrics import classification_report, confusion_matrix
import itertools

BATCH_SIZE = 32
```

```

IMAGE_SIZE = (320, 320)
# Walk through each directory
dataset = "sample_Signature/"
image_dir = Path(dataset)

# Get filepaths and labels
filepaths = list(image_dir.glob(r'**/*.JPG')) + list(image_dir.glob(r'**/*.jpg')) +
list(image_dir.glob(r'**/*.png'))

labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df = pd.concat([filepaths, labels], axis=1)
len(list(image_dir.glob(r'**/*.png')))

image_df

import matplotlib.image as mpimg
# Display 16 picture of the dataset with their labels
random_index = np.random.randint(0, len(image_df), 16)
fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(10, 10),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    image = Image.open(image_df.Filepath[random_index[i]])
    ax.imshow(image)
    ax.set_title(image_df.Label[random_index[i]])
plt.tight_layout()
plt.show()

# Separate in train and test data

```

```

train_df, test_df = train_test_split(image_df, test_size=0.2, shuffle=True,
random_state=42)
train_generator = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.resnet50.preprocess_input,
    validation_split=0.2
)

```

```

test_generator = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.resnet50.preprocess_input
)

```

Split the data into three categories.

```

train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='training'
)

```

```

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
)

```

```

        subset='validation'
    )

test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)

# Resize Layer
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(224,224),
    layers.experimental.preprocessing.Rescaling(1./255),
])

# Load the pretrained model
pretrained_model = tf.keras.applications.ResNet50(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

pretrained_model.trainable = False

# Create checkpoint callback
checkpoint_path = "fires_classification_model_checkpoint"
checkpoint_callback = ModelCheckpoint(checkpoint_path,
                                     save_weights_only=True,

```

```
monitor="val_accuracy",  
save_best_only=True)
```

```
# Setup EarlyStopping callback to stop training if model's val_loss doesn't improve  
for 3 epochs
```

```
early_stopping = EarlyStopping(monitor = "val_loss", # watch the val loss metric  
                               patience = 5,  
                               restore_best_weights = True) # if val loss decreases for 3  
epochs in a row, stop training
```

```
inputs = pretrained_model.input  
x = resize_and_rescale(inputs)
```

```
x = Dense(256, activation='relu')(pretrained_model.output)  
x = Dropout(0.2)(x)  
x = Dense(256, activation='relu')(x)  
x = Dropout(0.2)(x)
```

```
outputs = Dense(2, activation='softmax')(x)
```

```
model = Model(inputs=inputs, outputs=outputs)
```

```
model.compile(  
    optimizer=Adam(0.0001),  
    loss='categorical_crossentropy',  
    metrics=['accuracy']  
)
```

```
history = model.fit(  
    train_images,  
    steps_per_epoch=len(train_images),  
    validation_data=val_images,  
    validation_steps=len(val_images),  
    epochs=50,
```

```

callbacks=[
    early_stopping,
    checkpoint_callback,
]
)

results = model.evaluate(test_images, verbose=0)
model.save('model2.h5')
print("  Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))

# Predict the label of the test_images
pred = model.predict(test_images)
pred = np.argmax(pred,axis=1)

# Map the label
labels = (train_images.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred = [labels[k] for k in pred]

# Display the result
print(f'The first 5 predictions: {pred[:5]}')

# Display 25 random pictures from the dataset with their labels
random_index = np.random.randint(0, len(test_df) - 1, 15)
fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(25, 15),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    image = Image.open(test_df.Filepath.iloc[random_index[i]])
    ax.imshow(image)
    if test_df.Label.iloc[random_index[i]] == pred[random_index[i]]:
        color = "green"
    else:

```

```

    color = "red"
    ax.set_title(f"True: {test_df.Label.iloc[random_index[i]]}\nPredicted:
{pred[random_index[i]]}", color=color)
plt.show()
plt.tight_layout()

```

```

# Plot the accuracy and loss curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

```

```

epochs_range = range(len(acc))

```

```

plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

```

```

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

```

```

# Predict the label of the test_images
pred = model.predict(test_images)
pred_labels = np.argmax(pred, axis=1)
true_labels = test_images.labels

```

```

# Confusion Matrix
cm = confusion_matrix(true_labels, pred_labels)
print("Confusion Matrix:")
print(cm)

# Plot confusion matrix
plt.figure(figsize=(8, 8))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(test_images.class_indices))
plt.xticks(tick_marks, test_images.class_indices, rotation=90)
plt.yticks(tick_marks, test_images.class_indices)
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

```

app.py:

```

from flask import Flask, render_template, request
import os
from werkzeug.utils import secure_filename
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array

app = Flask(__name__)

```

```

UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def pred_class(pred):
    if pred == 0:
        return 'forged'
    elif pred == 1:
        return 'genuine'
    else:
        return 'unknown'

@app.route("/")
def index():
    return render_template('index.html')

@app.route("/classify", methods=['POST'])
def classify():
    image = request.files['file']A
    image.save(os.path.join(app.config['UPLOAD_FOLDER'],
secure_filename(image.filename)))

    # load the saved model
    model = load_model('model2.h5')

    # load and preprocess the new image
    img_path = os.path.join(app.config['UPLOAD_FOLDER'],
secure_filename(image.filename))
    img = load_img(img_path, target_size=(224, 224))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, 0)
    img_array = tf.keras.applications.resnet50.preprocess_input(img_array)

    # predict the label of the new image

```

```

predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=-1)

final_response = pred_class(predicted_class)

return render_template('classification.html',
image=secure_filename(image.filename), result=final_response)

if __name__ == '__main__':
    app.run(debug=True,port=5600)
from tensorflow.keras.models import load_model
import tensorflow as tf
import numpy as np

# Load the saved model
model = load_model('model2.h5')

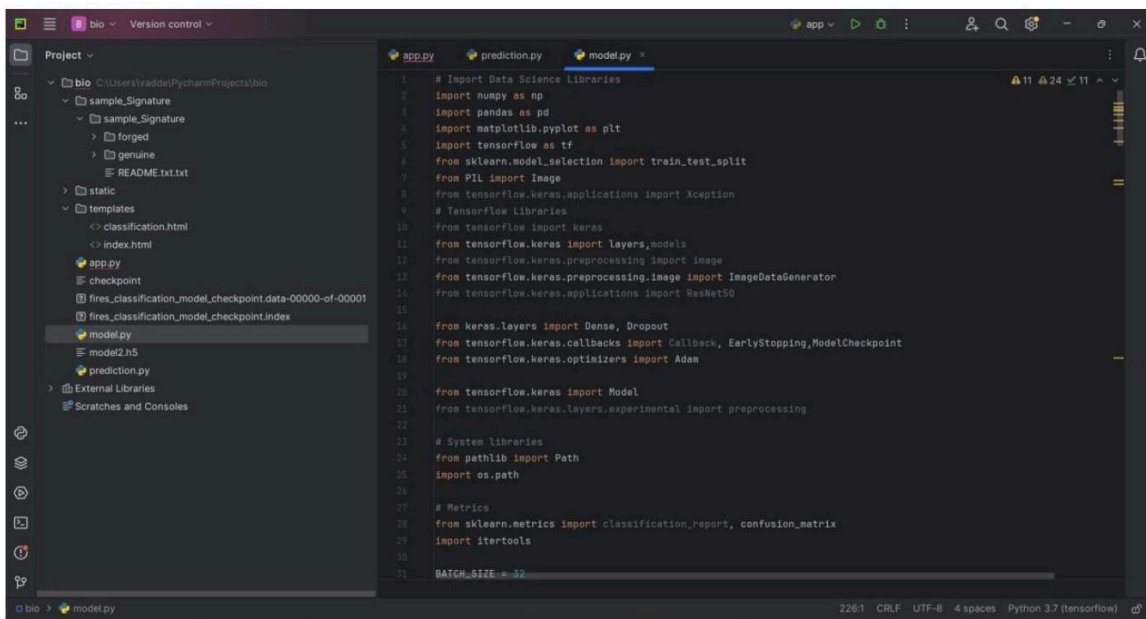
# Load and preprocess the new image
img_path = 'sample_Signature/sample_Signature/genuine/1 (12).png'
img = tf.keras.preprocessing.image.load_img(img_path, target_size=(224, 224))
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
img_array = tf.keras.applications.resnet50.preprocess_input(img_array)

# Predict the label of the new image
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=-1)

# Print the predicted label
if predicted_class == 0:
    print('Forged')
else:
    print('Genuine')

```

B. SCREENSHOTS



```
1 # Import Data Science Libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import tensorflow as tf
6 from sklearn.model_selection import train_test_split
7 from PIL import Image
8 from tensorflow.keras.applications import Xception
9 # Tensorflow Libraries
10 from tensorflow import keras
11 from tensorflow.keras import layers, models
12 from tensorflow.keras.preprocessing import image
13 from tensorflow.keras.preprocessing.image import ImageDataGenerator
14 from tensorflow.keras.applications import ResNet50
15
16 from keras.layers import Dense, Dropout
17 from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
18 from tensorflow.keras.optimizers import Adam
19
20 from tensorflow.keras import Model
21 from tensorflow.keras.layers.experimental import preprocessing
22
23 # System Libraries
24 from pathlib import Path
25 import os.path
26
27 # Metrics
28 from sklearn.metrics import classification_report, confusion_matrix
29 import itertools
30
31 BATCH_SIZE = 32
```

Fig. B.1: Importing Required Libraries



Fig. B.2: Dataset

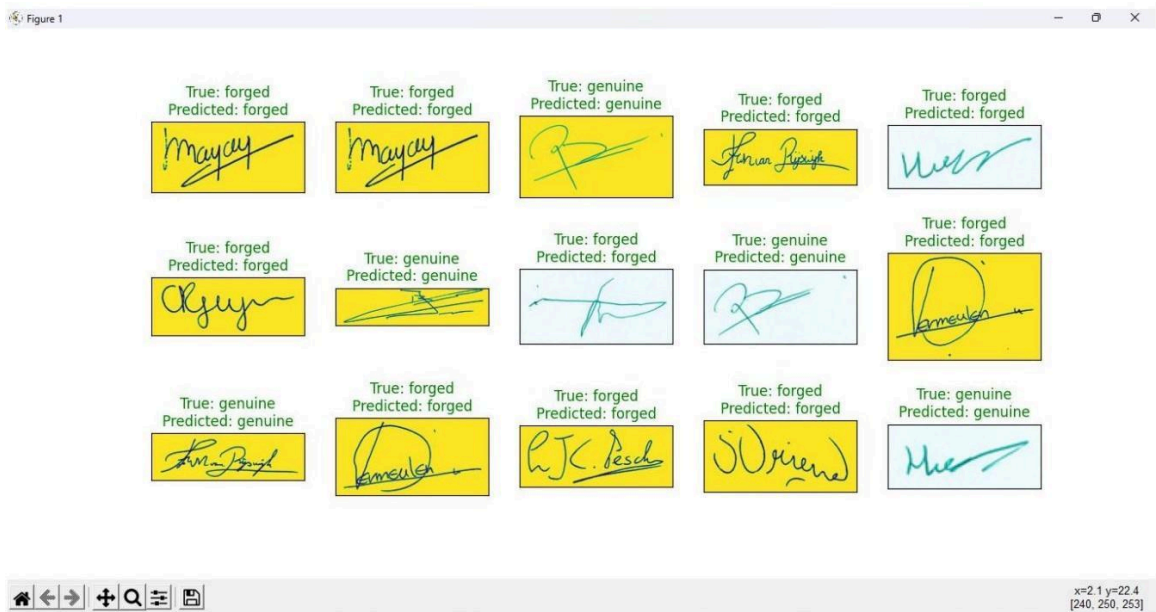


Fig. B.3: Dataset

```

Test Loss: 0.11945
Test Accuracy: 97.06%
7/7 [=====] - 19s 2s/step
The first 5 predictions: ['genuine', 'genuine', 'genuine', 'forged', 'genuine']
7/7 [=====] - 18s 2s/step
Confusion Matrix:
[[ 91   4]
 [  2 107]]

```

Fig. B.4: Model Training

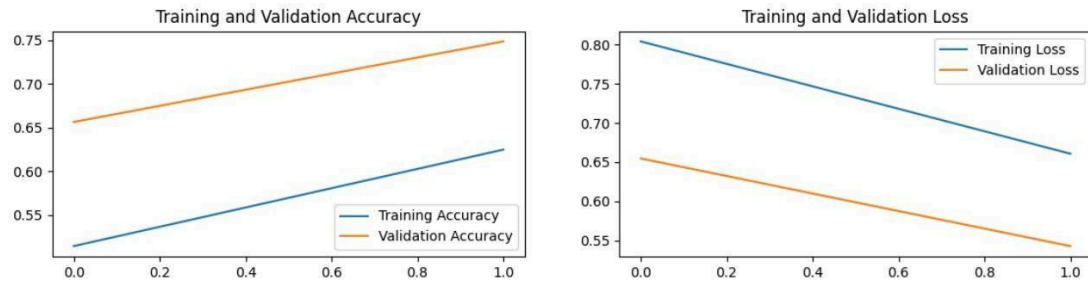


Fig. B.5: Performance Evaluation

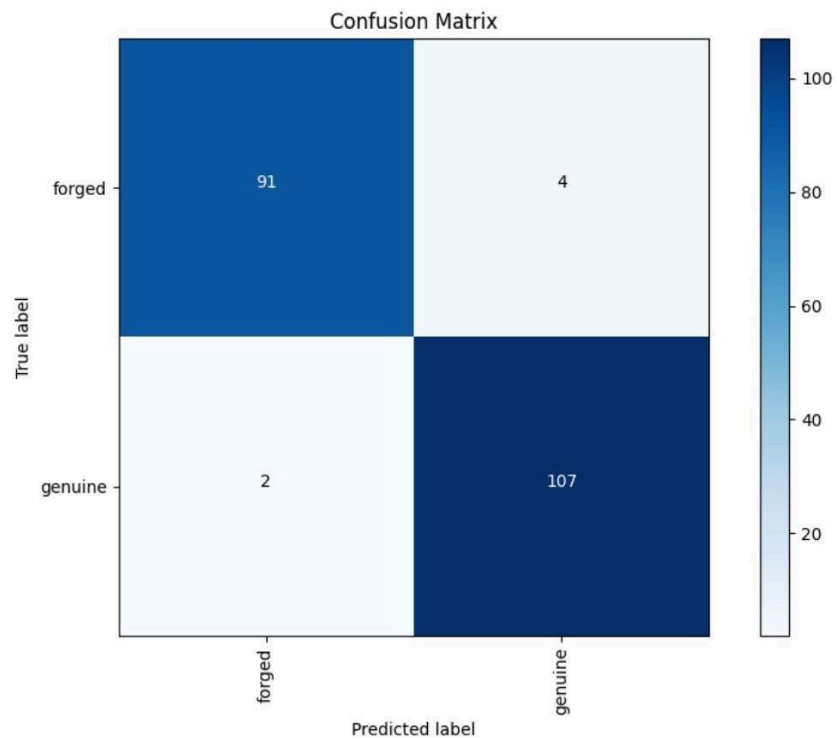


Fig. B.6: Confusion Matrix

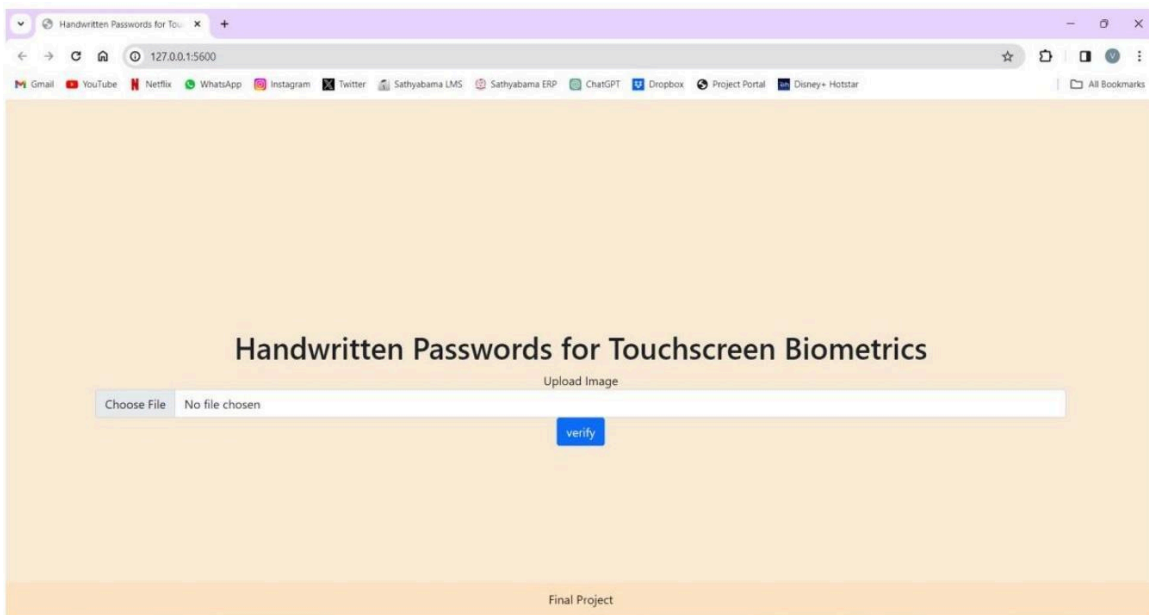
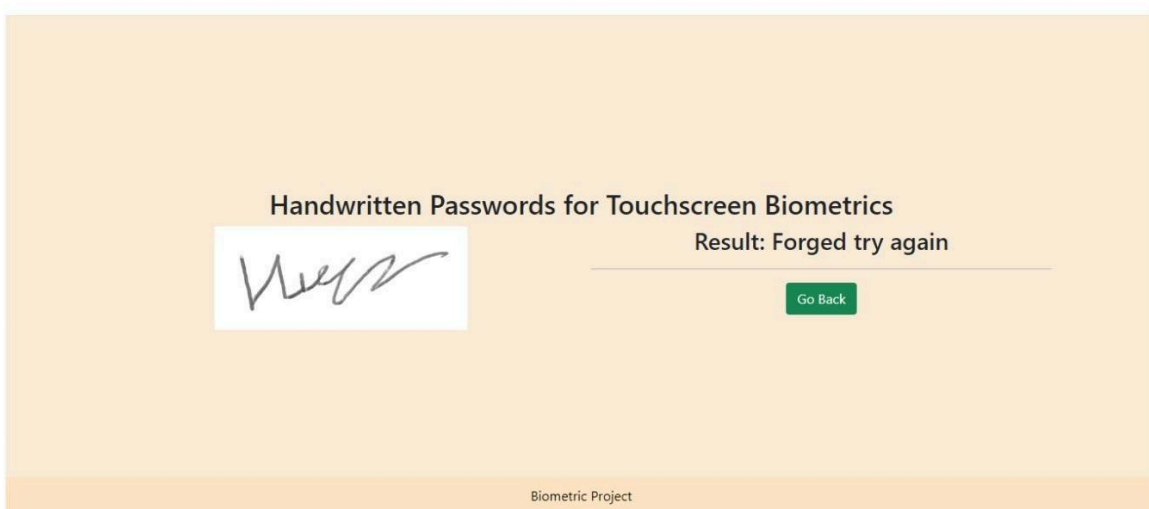


Fig. B.7: Flask Website



Handwritten Passwords for Touchscreen Biometrics

Vishal Reddy Vadde
*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
vaddevishalreddy198@gmail.com*

Yogitha.R
*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
yogith.cse@sathyabama.ac.in*

G.Kalaiarasi
*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
kalaiarasi.cse@sathyabama.ac.in*

M.Selvi
*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
selvi.cse@sathyabama.ac.in*

P Sardar Maran
*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
psmaran@sathyabama.ac.in*

Abstract— In the realm of biometric authentication, this project presents a novel approach that integrates the unique characteristics of handwritten signatures with touchscreen technology to establish a secure and user-friendly password matching system. The system leverages the power of Convolutional Neural Network (CNN) transfer learning, employing architectures like ResNet and DenseNet121, to recognize and authenticate handwritten signatures. A Flask web application serves as the interface, enabling users to submit their handwritten signatures for verification. Successful matches grant access to the device, showcasing the potential of this innovative biometric approach. The project commences with data collection and preprocessing, where a diverse dataset of genuine handwritten signatures is gathered from users. The Flask web application acts as the user-facing component of the system. Users are prompted to submit their handwritten signatures image input. The submitted signature is subjected to preprocessing to ensure consistency and optimal input for the trained model. The trained model then evaluates the signature's authenticity and similarity to the user's genuine signature. Upon a positive match, the application displays a "Password Matched" message and simulates device unlock, further underscoring the usability and real-world applicability of the. Finally, this project pioneers an innovative biometric authentication system that merges the familiarity of handwritten signatures with touchscreen technology.

I. INTRODUCTION

In an period marked by technological advancements and the proliferation of digital relations, the realm of security and authentication has evolved significantly. The confluence of touchscreen biometrics and transfer literacy through Convolutional Neural Networks(CNNs) forms the foundation of this design's invention. Touchscreens have come ubiquitous in ultramodern bias, enabling natural and intuitive relations. Integrating touchscreen- grounded hand input with biometric recognition promises a flawless stoner experience. Transfer literacy, a fashion that leveragespre-trained neural network

models to address specific tasks, empowers the system to learn the intricate nuances of individual autographs efficiently.

Every person has a unique signature which mostly includes the name of that person and there could be multiple people with the same name but that doesn't make their signature the same because everyone has their own way or style of writing. In the digital world, we use passwords and security questions to authenticate ourselves but in the physical world, we acknowledge ourselves by our identity cards like passports, driver's licence, etc. To make any kind of transaction we need those identification documents and our signature with it to because those are the things that verify us. Many people are determined to use our signatures and id to benefit from that because of that reason we need fraudulent signature detection software in place to identify and stop those phony activities. These types of signature forgery frauds are usually taking place in the legal processing or financial transactions like banking system where the person committing the fraud writes up a cheque and carefully signs it as the owner and tries to steal money but humans cannot tell if that is a genuine or forged signature just by seeing it that's where the fraud signature detection model comes to verify the signature and authenticate its legitimacy.

These fraudulent signatures can be detected in many ways like comparing them with a predefined datasets which is the most efficient way to detect whether a signature is genuine or forged to do that the predefined dataset should contain both the genuine and forged signatures so that the model can understand and tell the difference between the right signature and the wrong one and the other way is to physically analyze the hand signature style while signing which can be measured to know if there are any difference with the original one but this is not as promising when it compared with the other method.

In recent times, deep literacy ways have shown promising results in colorful image bracket tasks. Convolutional neural

networks(CNNs) have come particularly popular for image bracket and have been applied in colorful operations, including hand verification. One of the extensively used CNN infrastructures for hand verification is ResNet50.

In this design, we propose using ResNet50 for fake hand discovery. We aim to make a model that can directly identify whether a given hand is genuine or forged. We'll use a dataset of hand images, preprocess them to insure uniformity, and train the ResNet50 model using transfer literacy. We'll estimate how well the model is working by comparing it with the other models.

II. RELATED WORKS

1. Kshitij Swapnil Jain [1] The research objective is to develop a system for verifying the authenticity of signatures by distinguishing between genuine and forged ones. Recognizing the challenges posed by the evolution in handwriting and the skill of forgers, the study proposes the use of a Convolutional Neural Network (CNN) as a feature extractor and classifier. The CNN is employed to extract features through convolution filtering and down sampling. The hypothesis is that by training the CNN on a dataset of genuine and forged signatures, it can learn to differentiate behavioral characteristics, such as delays and hesitation, from the extracted data. To address the issue of vanishing gradients in deep networks during backpropagation, the study opts for a Residual Neural Network (ResNet). ResNet's ability to skip connections helps prevent the gradient from diminishing to insignificance, enhancing the training process. The proposed methodology aims to improve the efficiency and accuracy of forgery detection by leveraging the capabilities of CNNs and addressing gradient-related challenges through ResNet's architecture.

2. The research conducted by Kiran, Lakkoju Chandra [2], focuses on methods for authenticating visual media, especially the detection of manipulated images, with an emphasis on signatures, which are essential elements of authentication. The method is suitable for online and offline applications. Offline methods compare the current signature to reference signatures, while online methods use dynamic written information for high accuracy. The study implements a web-based classification system using a dataset containing 2000 images with both forged and genuine signatures. The process involves grayscale conversion of images, followed by binary conversion, denoising, and signature resizing. A .CSV file will be created containing extracted preprocessed data. A convolutional neural network (CNN) with three input layers, each with different weights and biases, and three hidden layers, takes a .CSV file as input and outputs the signature authentication. The optimizer and loss function are defined to minimize the loss and the Softmax layer calculates the accuracy. This comprehensive approach aims to improve the accuracy of verification of forged and genuine signatures in visual media.

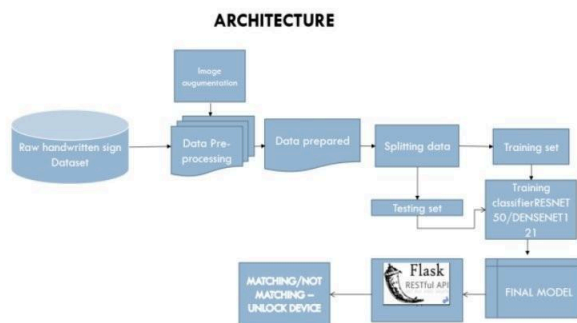
3. The paper proposed by Rashi Gundapwar, Aditi Parekh [3], a manual detection and control system based on CNN. The literature review covers several aspects of manual authentication. Various authors focus on static and dynamic

signatures, several offline hand inspection approaches, the impact of behavioral changes on individual hand characteristics, automatic hand inspection systems using image processing and machine learning advances, and the extraction of geometric parameters from preprocessed signatures. The purpose of the proposed system is to distinguish real and fake hands using a multi-step process. First, a dataset containing 750 autographs of 150 individuals is collected to create a database of unique characteristics. These signatures are then converted to black and white images using a grayscale algorithm, stored as matrices, and geometric transformations are performed. The preprocessed signature features are then extracted and compared with the features stored in the system. Finally, the CNN is used to determine the authenticity of the hand by comparing its features with those of a real hand. CNN contains several layers, including convolutional subcast, convolutional subcast, and fully connected subcast.

4. The paper presented by Belal A. M. Ashqar, Bassem S. Abu-Nasser [4], a system that uses deep learning techniques for manual inspection and forgery detection. The study distinguishes between two manual verification systems called offline and online which are also called static and dynamic. The system uses pattern matching and hidden Markov models for manual verification, while convolutional neural networks (CNNs) are used for fake detection. The proposed method focuses on changes in the behavior of handwritten signatures rather than intrinsic properties. Initially, signatures and their unique characteristics are collected to create a database. Autographs go through preprocessing, including resizing and splitting into two subfolders. The images are then loaded into a CNN implemented with Keras and a TensorFlow backend, and the system performance is evaluated. The article discusses CNN operations including convolution, max pooling, fully connected subboxing, ReLu (rectifier), and softmax. The database used contains 300 images, 150 genuine and 150 fake images of 30 individuals with genuine autographs. The methodology produces material with an 82% distribution rate, achieving high sensitivities ranging from 99.7% to 99.9%. The conclusion indicates the successful use and effectiveness of the proposed method in manual inspection and detection of counterfeits.

5. The paper proposed by Jivesh [5], a comprehensive system for hand detection, forgery detection, and authentication that uses CNN, Crest-Through method, SURF algorithm, and Harris corner detection algorithm. The methodology includes literature reviews and autographs are processed using CNN and Crest-Through method. The use of CNN and Crest-Through methods are for manual detection and authentication, which include functions such as denoising, scaling, centering, rotation, length/aspect, area/aspect, and adjustment of Crest-Through parameters. Both CNN and the Crest-Through Method contribute to hand recognition. For forgery detection, the Harris corner detection algorithm is used to detect and estimate the corner points, while the SIFT algorithm detects and analyzes the indicator points to compare them with genuine hands.

SYSTEM ARCHITECTURE:



III. PROPOSED SYSTEM

MODULE-1: DATA COLLECTION

The data used to train the model is collected from 30 different people where each person's signature is taken 10 times (5 from themselves and 5 from others), the 5 signatures taken by the right person are considered as genuine signature and the other 5 are considered as forged signatures. The naming of the signature are done accordingly: the naming of each signature is divided into two parts of three digit number, if the first part and last part of the number is same then the signature is done by themselves if the first the first part is different from the last then the signature is done by different person.

MODULE-2: DATA PREPROCESSING

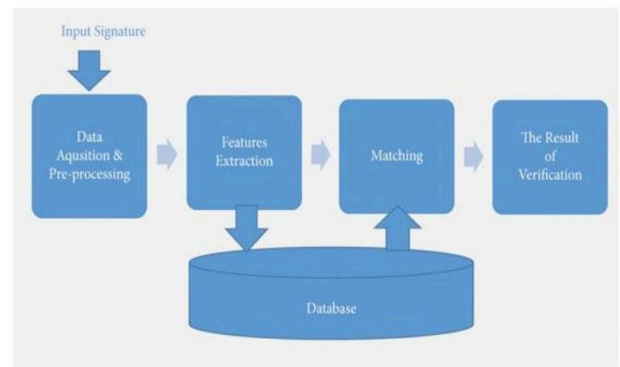
Choosing a dataset is a big task while working with deep neural networks because the data which is used to train the model should be clean to make it clean the dataset goes under a series of steps to make the images clear and to make sure those images are ready to be used to train the model.

Data preprocessing is nothing but getting the dataset ready for the model on which the model analyzes, learns and gains the ability to recognize the inputs or the objects.

Every image in the dataset have some sort of distortion or lacks enough clarity which will be corrected in the process. Using Keras the images will be flipped upside down, resized and enhanced so that the model can analyze the better parts of the image.

Image Augmentation is the concept well known for data preprocessing. To tackle the necessity of cultivating a resilient network capable of handling these diverse elements, the adoption of Data Augmentation emerges as a crucial strategy.

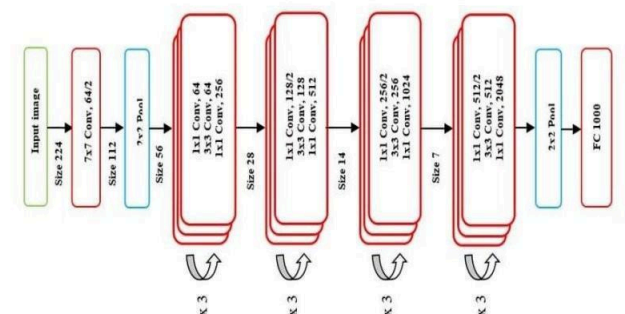
BLOCK DIAGRAM:



MODULES -3: MODEL TRAINING

ResNet50 stands out as a cutting-edge deep learning model, particularly excelling in diverse computer vision applications. Its architecture incorporates a key innovation known as residual connections, allowing the model to learn from both the original and residual images. This distinctive design empowers the network to grasp more intricate features and effectively mitigates the challenge of vanishing gradients during training. In the specific context of detecting forged signatures, ResNet50 proves to be a potent tool for identifying subtle differences between genuine and forged signatures. The model autonomously learns crucial features for signature verification, including the nuanced shapes and structures of signature strokes. This capability sets ResNet50 apart from traditional machine learning approaches, which may struggle to accurately capture these intricate signature characteristics. An additional strength of employing ResNet50 for fake signature detection lies in its compatibility with transfer learning. Leveraging this technique involves utilizing a pre-trained ResNet50 model, often on expansive datasets like ImageNet, as a starting point for a new task, such as signature verification. The use of transfer learning enables the model to inherit and fine-tune features learned during its pre-training phase, even when confronted with a relatively small dataset. This adaptability and transferability make ResNet50 an advantageous choice for signature verification tasks, showcasing its versatility and robust performance in the realm of deep learning applications.

ResNet50 Architecture:



IV. PERFORMANCE EVALUATION

MODEL TESTING:

Classification Accuracy:

$$Accuracy = \frac{\text{Number of Correct predictions}}{\text{Total number of predictions made}}$$

Logarithmic Loss:

$$LogarithmicLoss = \frac{-1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} * \log(p_{ij})$$

Where, Y_{ij} tells if i belong to j or not

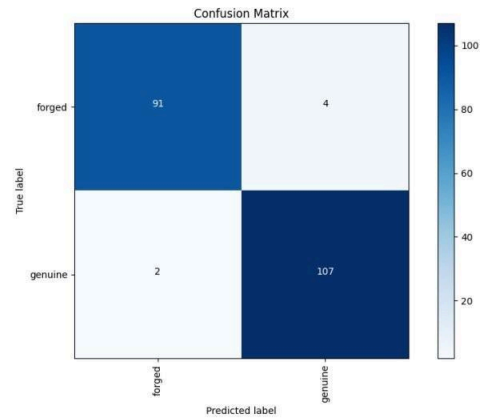
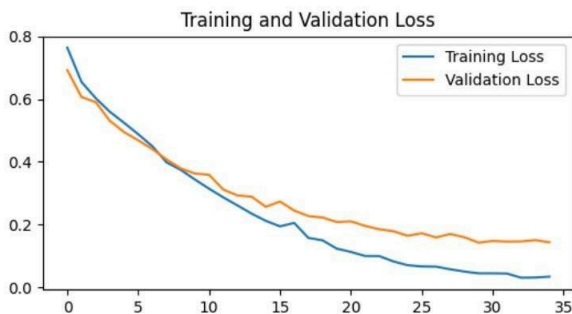
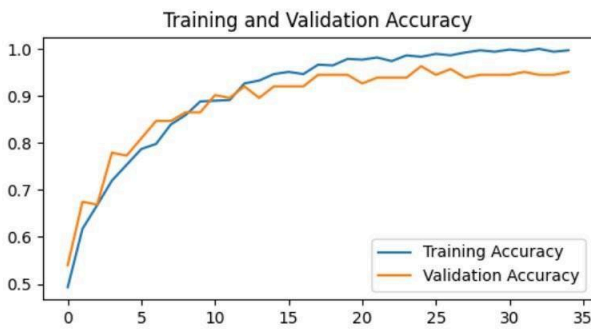
P_{ij} tells the probability of i belonging to j

Log Loss has the range $[0, \infty)$.

If Log Loss is near zero then it probably is at the highest accuracy and if it is away from zero then lower the accuracy.

V. RESULT AND DISCUSSION

The idea of using the Convolutional Neural Network (CNN) transfer learning, employing architectures like ResNet50 was a revolutionary concept that made the model achieve 97% accuracy and validation loss of less than 1%.



According to the confusion matrix after the completion of model training the number of false negative for the genuine is just two and the false negative for forged is four which tells us the model was successful in determining the correct signatures.

VI. CONCLUSION

In a world full of cyber-attacks and the constantly increased use of technology on our daily basis we need to protect our privacy which is a hard thing but we can try new ways to protect our data. The passwords that are unique to everyone helps in do that. Our handwritten signatures can be one of the ways to add extra protection. The implementation of Convolutional Neural Networks and transfer learning make sure for to use our handwritten passwords to protect our data. The developments in deep learning in the future could definitely take this to the next level of neural networking which makes the identification forged signatures easier.

REFERENCES

- [1] Kiran, Lakkoju Chandra, Gorantla Akhil Chowdary, Manchala Shalem Raju, and Kondaveeti Gopi. "Digital signature Forgery Detection using CNN." International Research Journal of Engineering and Technology 8, no. 6 (2021): 2969-2973.
- [2] Jain, Kshitij Swapnil, Udit Amit Patel, and Rushab Kheni. "HANDWRITTEN SIGNATURES FORGERY DETECTION." (2021).
- [3] Subramaniam, M., Teja, E. and Mathew, A., 2022. Signature forgery detection using machine learning. Int. Res. J. Modernization Eng. Technol. Sci., 4(2), pp.479-483.
- [4] Alajrami, Eman, Belal AM Ashqar, Bassem S. Abu-Nasser, Ahmed J. Khalil, Musleh M. Musleh, Alaa M. Barhoom, and Samy S. Abu-Naser. "Handwritten signature verification using deep learning." (2020).
- [5] 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).
- [6] Poddar, Jivesh, Vinanti Parikh, and Santosh Kumar Bharti. "Offline signature recognition and forgery detection using deep learning." *Procedia Computer Science* 170 (2020): 610-617.
 - [7] Jain, S., Khanna, M. and Singh, A., 2021, February. Comparison among different cnn architectures for signature forgery detection using siamese neural network. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (pp. 481-486). IEEE.
 - [8] Navid, Shayekh Mohiuddin Ahmed, Shamima Haque Priya, Nabiul Hoque Khandakar, Zannatul Ferdous, and Akm Bahalul Haque. "Signature verification using convolutional neural network." In *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, pp. 35-39. IEEE, 2019.
 - [9] Rateria, Avani, and Suneeta Agarwal. "Off-line signature verification through machine learning." In *2018 5th IEEE Uttar Pradesh section international conference on electrical, electronics and computer engineering (UPCON)*, pp. 1-7. IEEE, 2018.
 - [10] Foroozandeh, Atefeh, Ataollah Askari Hemmat, and Hossein Rabbani. "Offline handwritten signature verification and recognition based on deep transfer learning." In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pp. 1-7. IEEE, 2020.