International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-11 Issue-4, April 2022

Performance Analysis of Deep Learning Approaches for Offline Signature Verification

Mallikarjuna Gundavarapu Rao, Kompelly Sreeja, Sai Anoushka, Telugu Akila, Sneha Nimmala

Abstract: ***Signature verification plays a significant role in many biometric authentication places. Many financial institutes require a robust signature verification process for check clearance, loan sanction, processing, pension relation documents, etc. Expert forgeries make it hard to authenticate an individual's identification based on signatures. Typically, this occurs when the forger understands the user's intricate features of the signature and strives to mimic it. Online signature verification approaches can extract various features such as keystrokes, pressure of the pointer, duration between the strokes and the lettering styles, so that verification becomes effective. However, the lack of these intricate details in offline signature, the authentication process becomes much more difficult. To address these issues, in this paper we propose deep learning-based approaches for offline signature verification. In this regard, we have used ZFNet, LeNet and AlexNet architectures with CEDAR, BHSig20 and UTsig datasets for our extensive. experimentation. We propose a learning model in which the dataset consists of multiple genuine and forged signatures. Further, performance analysis of these techniques has been carried out. It was found that LeNet has provided better training and testing accuracy with above 82% performance.***

Keywords: ZFNet, LeNet, AlexNet, Writer- Independent detection, CNN, Signature Verification System.

I. INTRODUCTION

The most basic form of identifying a person is their signature, consisting of strokes, shapes, and formats. This signature can authorize a cheque or document or conclude a letter as a proof of distinctiveness and intent. It serves as verification of a person's identity. A signatory or signer is the individual who makes a signature. Signatures have been an intriguing component of human civilization for a very long time. In our modern age, biometrics are used in every field for security purposes. A framework like this analyses a person based on physiological or behavioural features. Estimates of natural evidence qualities, such as the unique finger impression, face, exclusive iris, and so on.

Manuscript received on 25 March 2022.
Revised Manuscript received on 31 March 2022.
Manuscript published on 30 April 2022.

* Correspondence Author
Mallikarjuna Rao Gundavarapu, Professor, Department of Computer Science, GRIET, Hyderabad (Telangana), India E-mail: gmallikarjuna628@grietcollege.com
Sreeja Kompelly, B.Tech Student, Department of Computer Science, GRIET, Hyderabad (Telangana), India. E-mail: srijakompelly@gmail.com
Sai Anoushka Kokku, B.Tech Student, Department of Computer Science, GRIET, Hyderabad (Telangana), India. E-mail: sai.anoushka30@gmail.com
Akila Telugu*, B.Tech Student, Department of Computer Science, GRIET, Hyderabad (Telangana), India. E-mail: teluguakila307@gmail.com
Sneha Nimmala, B.Tech Student, Department of Computer Science, GRIET, Hyderabad (Telangana), India. E-mail: nimmalasneha@gmail.com

Behavioural characteristics such as handwritten signature and voice are the latter part. In these two signatures, biometric frameworks are often used.
Verification: In this situation, a biometric sample is provided by the user and confirms their identification; the verification system validates the user's authenticity. Identification: The major goal in this situation is to identify the individual from the provided batch.
The signature verification system will automatically assess whether a biometric sample belongs to a claimed individual. Forgeries are classified into three kinds:

1. Random forgeries: The unexpected forgery cases are in the imitator without any data about the client or signature data. The imitator Utilizes their signature to submit fabrication. The delivered sign differs in meaning and has a distinct style. Thus, these are the easiest to identify.
2. Simple forgeries: Simple forgeries are the forgeries where the forger has an uncertain idea regarding the name; however, it is without the signature format. This case might look a little like the authentic signature; for most, those cases were in the clients' full signatures.
3. Skill forgeries: Skill forgeries are those where the imitator can figure out the client's name used in the sign as well as the format of the client sign. They observe and focus on consistently repeating a similar mark. These forgeries are extremely accomplished and nearly identical to real signatures, and hands are difficult to distinguish. Signature verification frameworks are of two types:

1. Online framework (dynamic): Whenever the signature is delivered, the pointer's movement is considered in the online technique. The algorithms employed here also look at the position, pressure used by the pen, speed of signature, and acceleration. These dynamic features are Distinctive to every person, and they're also rehashed Throughout the entire time duration.
2. Offline framework (static): The technique utilised in offline signature verification is considerably different since the signature analysis can only be performed after the signature has been completed. The signature is shown here using computerised digital graphics. A system for verification of signature (SVS) assists an organisation in determining the validity of a wide range of legal documents. A handwritten signature is the simplest and most secure way of identification. Signatures have played an important role in a variety of industries, including financial, economic, and legal activities.

II. LITERATURE SURVEY

The paper by M.Abadi et al. [1] used TensorFlow for large scale ML(machine learning), while the paper by H.

Baltzakis et al. [2] used a two-stage perceptron OCON (one-class –one-network) for every global feature set extracted from grid and texture. The first stage extracts decisions from feature data while the radial base function, RBF, of the second stage performs classification. Most of the authors employed OpenCV and Keras in the python environment et al. [3,4]. They claimed a higher accuracy rate for recognition. Dutta et al. [5] used local associative indexing for verification of signature that is writer independent. They used HOG (Histogram of Oriented Gradients) for the hybridization of local features and the global statistical features that are obtained from the signature image. The spatial information between local features has been used for separating forged and original signatures. Here, learning a condensed set of neighbouring features (that are of higher order) based on visual words, like doublets and triplets, is a challenging problem due to the associated exponential time complexity. DWT (Discrete Wavelet Transform) and the Image Fusion method has been used by S. Ghandali et al. [6] for offline Persian signature identification and verification. Experimental results demonstrate the effectiveness of the proposed method in classifying the images based on the Euclidean distance between the test image and each pattern. K. Han and I. K. Sethi et al. [7] used a set of geometrical and topological elements to map the signature image into two strings of the local associative indexing to accept the partial input queries and prove the tolerance of the missing features. We have checked our system performance with an image database having 120 pictures. K. Huang et al. [8] introduced a technique for (offline) signature verification taking into regard the features that are extracted from images. In this technique, the shape of the signature plays a major role. Based on the similarity measure of shapes of signatures, a signature is recognized and verified. Similarly, a Neural network classifier is used to analyze the geometrical features of a signature image. And all the outputs from each scale are aggregated to obtain the final or overall match rate. The model is trained on genuine and forged samples that are artificially generated from enrollment referenced signatures. This allows training control and reduces the sample number that is required to achieve better performance. Oliveira and Otsu [9,10] adapted geometrical features and graphology for signature confirmation. Smart tools are presented for web investigation by RAO et. all.,[11] so that signatures can also be extracted from cloud medium..

III. EXISTING APPROACHES

The field of verifying offline signatures has been researched for the past few decades. The issue has been addressed from a variety of perspectives throughout the long period. A few of the existing methodologies are-

1. Earlier, professional workers were hired to compare the signatures on government documents to prove the authenticity of the signatures. However, this is highly unreliable and time taking.
2. There are many existing frameworks for offline signature verification; one such system utilizes feature extraction. However, it uses feature extraction where the model is prepared to extract features like slanted/skewness, kurtosis and eccentricity manually. This system is easier to understand and compare but couldn't perform well in detecting skilled forgeries.

IV. PROPOSED SYSTEM

In the proposed system, we are using CNN (Convolutional Neural Networks) architectures to train our model. Here we need not calculate features and identify the similarities, We train our model with CNN upon which the model maps out the patterns and logic between the training and testing dataset to validate the accuracy of the signature verification.

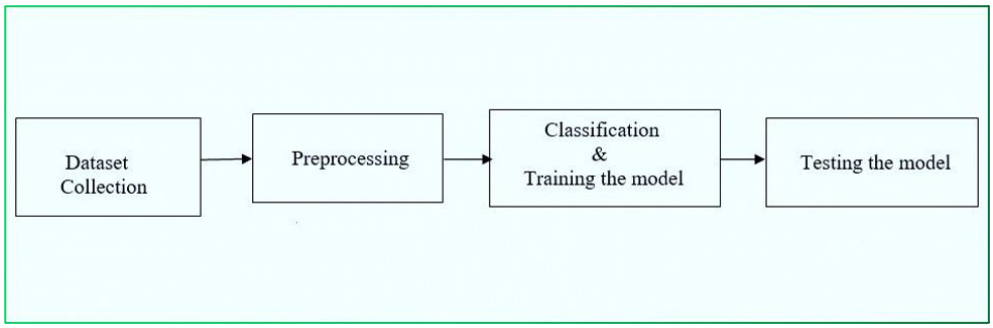


Figure-1: This Figure Represents the Four Modules in Our Proposed Approach.

Figure1, figure 2 and figure 3 give the architecture of the proposed approaches. CNN (Convolutional Neural Networks) are commonly utilised in domains involving the analysis of visual information; hence, these designs are well suited to signature verification. For high input sizes, this design scales better than completely connected models since it has fewer trainable parameters. Instead of employing feature extractors, we train the model with ZFNet, LeNet, and AlexNet architectures, which contain layers such as max pooling, flattening layers, Conv2D layers, and so on. **ZFNet:** The convolutional neural network ZFNet is a classic convolutional neural network. Visualizing intermediate feature layers and the classifier's operation inspired the design. The filter widths and stride of the convolutions are lowered compared to AlexNet. In comparison to AlexNet, the authors of ZFNet increased the number of filters in all convolutional layers and the number of neurons in the fully connected layers, in addition to modifying the filter size. The ZFNet has 96-256-384-384-256-4096-4096 kernels/neurons. This allowed the network to raise the complexity of internal representations, lowering the mistake rate from 15.4 percent last year to 14.8 percent this year.

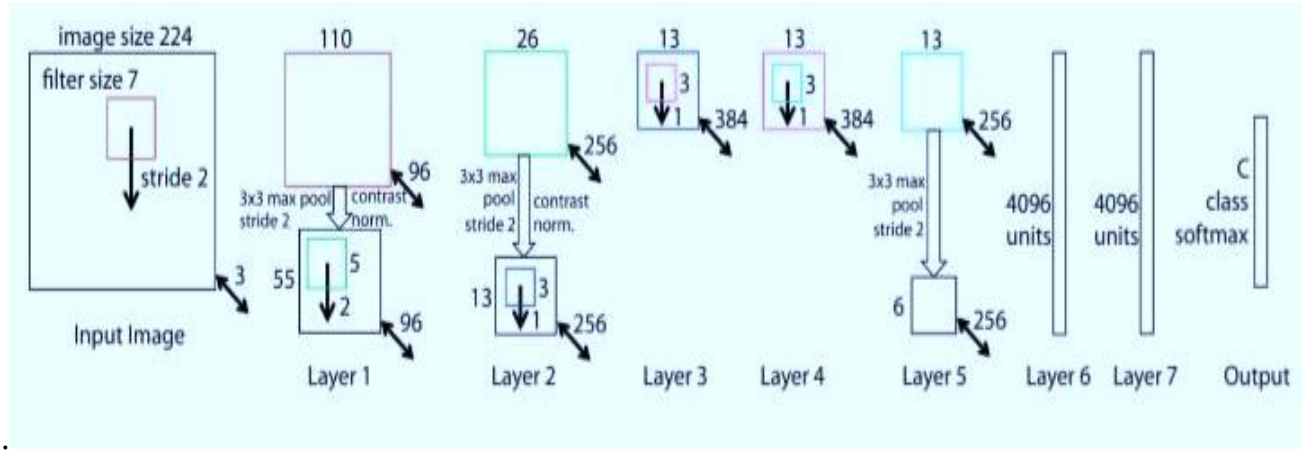


Figure-2: This figure represents the architecture of the ZFNet.

Table-I: This table represents the output upon building the ZFNet model.

Layer(Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 74, 74, 64)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
flatten (Flatten)	(None, 41472)	0
dense (Dense)	(None,128)	5308544
dense_1 (Dense)	(None, 2)	258
Total params: 5,329,058		
Trainable params: 5,329,058		
Non-trainable params: 0		

LeNet: LeNet-5 is one of the very first convolutional neural networks, and it aided deep learning research. LeNet, as an early convolutional neural network, possesses the basic units of convolutional neural networks, such as the convolutional layer, pooling layer, and complete connection layer, establishing the groundwork for future convolutional neural network development. As illustrated in the diagram (32*32-pixel input image data): There are seven layers in LeNet-5. Every other layer, in addition to input, can train parameters. Cx stands for convolution layer, Sx for sub-sampling layer, Fx for complete connection layer, and x for layer index in the diagram.

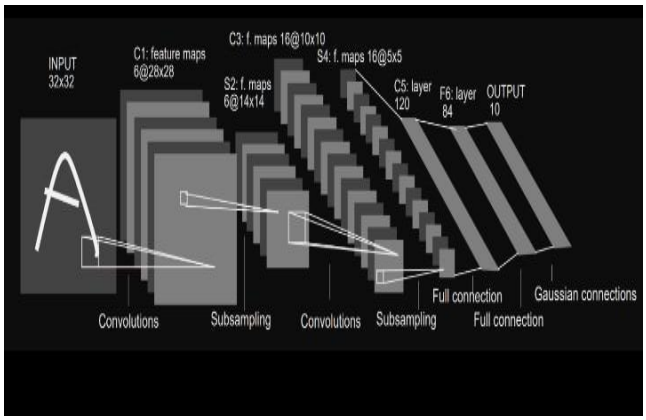


Figure 3: This figure represents the architecture of LeNet-5.

Table-II: This table Represents the Output Upon Building the LeNet model.

Layer(Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 6)	456
average_pooling2d (AveragePooling2D)	(None, 112, 112, 6)	0
conv2d_1(Conv2D)	(None, 108, 108, 16)	2416
average_pooling2d_1 (AveragePooling2D)	(None, 54, 54, 16)	0
flatten (Flatten)	(None, 46656)	0
dense (Dense)	(None,120)	5598840
dense_1 (Dense)	(None, 10)	1210
Total params: 5,602,922		
Trainable params: 5,602,922		
Non-trainable params: 0		

AlexNet: AlexNet has eight layers: the first five were convolutional layers, followed by max-pooling layers in some cases, and the final three were fully linked layers. [2] It used the non-saturating ReLU activation function, which outperformed tanh and sigmoid in terms of training performance.

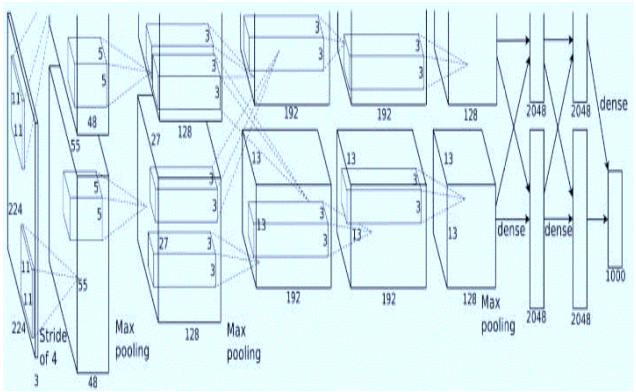


Figure 4: This figure represents the architecture of the AlexNet

Table-III: This Table Represents the Output Upon Building the AlexNet model

LAYER (TYPE)	OUTPUT SHAPE	PARAMETER #
conv2d_20 (Conv2D)	(None, 54, 54, 96)	34944
batch_normalization_20 (BatchNormalization)	(None, 54, 54, 96)	384
max_pooling2d_12 (MaxPooling2D)	(None, 26, 26, 96)	0
conv2d_21 (Conv2D)	(None, 26, 26, 256)	614656
batch_normalization_21 (BatchNormalization)	(None, 26, 26, 256)	1024
max_pooling2d_13 (MaxPooling2D)	(None, 12, 12, 256)	0
conv2d_22 (Conv2D)	(None, 12, 12, 384)	885120
batch_normalization_22 (BatchNormalization)	(None, 12, 12, 384)	1536
conv2d_23 (Conv2D)	(None, 12, 12, 384)	1327488
batch_normalization_23 (BatchNormalization)	(None, 12, 12, 384)	1536
conv2d_24 (Conv2D)	(None, 12, 12, 256)	884992
batch_normalization_24 (BatchNormalization)	(None, 12, 12, 256)	1024
max_pooling2d_14 (MaxPooling2D)	(None, 5, 5, 256)	0
flatten_4 (Flatten)	(None, 6400)	0
dense_12 (Dense)	(None, 4096)	26218496
dropout_8 (Dropout)	(None, 4096)	0
dense_13 (Dense)	(None, 4096)	16781312
dropout_9 (Dropout)	(None, 4096)	0
dense_14 (Dense)	(None, 10)	40970
Total Parameters:	46, 793, 482	
Trainable Parameters:	46, 790, 730	
Non-trainable Parameters:	2, 752	

V. DATASET

We have used this on four datasets for performance analysis, out of which 3 are training datasets. They are CEDAR, BHSig260, UTSig.

1. CEDAR: This dataset has a total of 55 signatures, and the people signed are from different social and expert backgrounds. Here we will take 24 genuine and 24 forged signatures into consideration for each user.

2. BHSig260: This dataset contains a total of 260 signers. The signers are Hindi and Bengali. There are 160 signers from Hindi and around 100 signers from Bengali. Each user has 30 forged signatures and 24 real signatures.

3. UTSig: This dataset contains signatures from the Persians. There are a total of 8320 signatures on this document. A total of 115 signatures are collected. And we have 45 fraudulent signatures and 37 real autographs for each individual. Fig 4 and fig 5 are sample images of real and forged signatures.

Figure 5: Sample image of a forged signature in the data set.

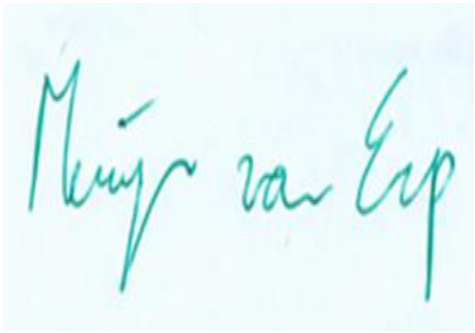


Figure 6: Sample image of a real signature in the data set.

VI. DEVELOPMENT FRAMEWORK

A. Software Installation

1) Visit the following website:

Colab.research.google.com

2) Sign in to collabs by using a google account.

3) To start a new runtime instance, select a new notebook. By clicking on it, you may modify the name of your notebook to whatever you like.

4) You enter your code in the cell execution block. A cell can be executed in two ways:

a) Press shift+enter

b) Click on the run symbol at the top left corner of each cell. You can also run all cells at a time by clicking "run all" in Runtime. Output for each cell is displayed right below the cell.

5) Colab allows users to upload files from their Google Drive account to access training data stored there. There are two ways to mount a drive in collab:

a) ***Using GUI*:** To mount your Google Drive, click on the file icon on the left side of the screen and then on the "Mount Drive" icon.

b) ***Using code:*** Execute the following code given below:

from google. colab import drive
.mount('/content/drive')

6) A pip is the package manager that is used to install packages. For example, run this command to install a certain version of TensorFlow:

!pip3 install tensorflow==1.5.0

The exclamation point instructs the notebook cell that the command should be executed as a shell command.

7) You can download your code as .ipynb or .py files. You can also save your code with outputs as pdf by clicking 'Print' in the 'File' icon and then choosing the "Save as pdf" option.

VII. EXECUTION

1. *Image input*: The program's execution begins with the display of examples of real and faked images. Then we read the paths of all the genuine and forged images. Here, we use the glob function, which is used to return all file paths that match a specific pattern. For accurate results, it's better to use good resolution images.

2. *Image loading*: The OpenCV module in Python contains a read function for reading an image from a provided path. The picture should be present in the same directory as the application, or the exact method of the image's information, such as the location, must be supplied as input. Cv2.imread is used to read the path of the specified image.

Img1=cv2.imread("Input_path_of_the_image")

3. Image preprocessing: The input picture is preprocessed at this step. The image is transformed in a variety of ways, including resizing and grey scaling. The OpenCV functions cat colour and resizes() are used for grey scaling and resizing, respectively. The dataset comprises a training dataset and a testing dataset. And the labels are given for genuine and forged images. Genuine images are labelled as '0', and forged images are labelled as '1'.

4. CNN model training: Before we proceed, we must train the CNN model to determine if a picture is authentic or counterfeit. During this step, we will create a neural network model. We create and train the CNN model with CNN layers and the fit technique using the Keras library.

Using the add () technique, we will slowly build a sequential model. The Conv2D class is used to create a basic CNN. The CNN model has been constructed. The preprocessed pictures are now sent into the CNN model to be trained.

5. Output: Testing is done following the training phase. Here the model will be tested against the testing dataset, and the output is displayed in terms of percentage, which is the accuracy percentage.

C. Result and Discussion

The application is finished and ready. In the application, the image paths are read first. Accuracy is enhanced by high-resolution photos. After reading the picture, it is preprocessed, which includes scaling using the resize () method, grayscale conversion with cvtColor, and so on. We used four datasets. Out of these, three are used as training datasets, and one is used as a testing dataset.

Table-IV: Experimental results: Performance Comparison of Deep Learning Networks

S.NO	EPOCHS	BATCH SIZE	ACTIVATION FUNCTIONS	Accuracy % in ZFNet	Accuracy % in LeNet	Accuracy % in AlexNet
1	5	1	Relu,softmax	72.7	50	57.2
2	25	32	Relu,softmax	76.6	69.4	52.7
3	100	64	Relu,softmax	76.1	54.4	51.6
4	50	32	Relu,softmax	77.2	82.2	49.4

The CNN model is built and then trained using the training dataset, which is further validated using the test dataset, and the final output is given in the form of a percentage, which gives us the accuracy. This accuracy tells us how accurately our model determines if an image is real/forged. The table IV gives the overall analysis of the accuracies obtained upon using three different architectures ZFNet, LeNet and AlexNet which can be used for comparison of the architectures. And for each architecture, analysis is also performed by changing epochs and batch size.

D. Pseudo code

```
//load the image from specified input path
 image = cv2.imread(i)
//converting image into grayscale
 image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
//resizing the image
image = cv2.resize(image, (224, 224))
//label genuine as 0 and forged as 1
train_labels.append(0) //for genuine
CNN model
//build the CNN model
//final output is displayed in percentage
format after testing.
```

Figure 6: Pseudo Code for The Frame Work

VIII. CONCLUSION AND FUTURE SCOPE

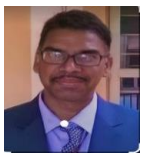
All through the last ten years, there have been numerous techniques for Offline Signature Verification. While correcting the difference between genuine and forged signatures, misleading acknowledgement rates have dropped fundamentally in the last several years considering movements in Deep Learning. We gave a lot of information to help you learn the signatures. The results reveal that Convolutional Neural Networks are superior for signature characterisation and understanding visual information differences between certifiable and fabricated signatures. The primary improvement in the accuracy and exactness occurred. Systems research must concentrate on broadening the scope of frameworks to accommodate more considerable variations encountered later. For instance, signatures endorsed in more modest spaces, in a rush, or on reports with meddling lines.

1. A component of our future effort will be developing a simpler and better score normalisation mechanism. Accessing a signature's complexity level can help with various concerns, such as client-based score normalisation or security needs.
2. This can also be further implemented by adding extra layers of security while authenticating the signatures to increase the accuracy and safety of systems predictions. It would be effortless if we could improve the process of giving inputs by scanning with a camera rather than uploading files.

REFERENCES

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensorflow:Enormous scope AI on heterogeneous disseminated frameworks," CoRR, vol abs/1603.04467, 2016.
2. H. Baltzakis and N. Papamarkos, "A new signature verification technique based on a two-stage neural network classifier", Engineering Applications of Artificial Intelligence, vol. 14, no. 1, pp. 95 – 103, 2001.
3. G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision in C++ with the OpenCV Library, 2nd ed. O'Reilly Media, Inc., 2013.
4. F. Chollet et al., "Keras (2015)," 2017.
5. A. Dutta, U. Pal, and J. Leads, "Compact correlated features for writer independent signature verification," in 2016 23rd International Conference on Pattern Recognition (ICPR), Dec 2016, pp. 3422–3427.
6. S. Ghandali and M. E. Moghaddam, "A method for off-line Persian signature identification and verification using dwt and image fusion," in 2008 IEEE International Symposium on Signal Processing and Information Technology, Dec 2008, pp. 315–319.
7. K. Han and I. K. Sethi, "Handwritten signature retrieval and identification," Pattern Recognition Letters, vol. 17, no. 1, pp. 83 – 90, 1996.
8. K. Huang and H. Yan, "Off-line signature verification based on geometric feature extraction and neural network classification," Pattern Recognition, vol. 30, no. 1, pp. 9 – 17, 1997.
9. L. S. Oliveira, E. Justino, C. Freitas, and R.Sabourin, "The graphology applied to signature verification" pp.286–290.
10. N. Otsu, "A threshold selection method from grey-level histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, Jan 1979.
11. Rao, Gundavarapu & Reddy, B. & Vishnu, Peesapati,"Smart Web Investigation Framework", Innovations in Cyber Physical Systems, 2021,pp.305-314, doi. 10.1007/978-981-16-4149-7_27

AUTHORS PROFILE



Mallikarjuna Rao Gundavarapu, G. Mallikarjuna Rao, working as a professor in CSE Department of Gokaraju Rangaraju Institute of Engineering and Technology. He is having rich academic experience of 32 years. His interesting domains are Pattern Recognition, IoT and Soft computing. He is the instrument in setting up " Parallel Computing and Operating Systems Lab" under MODROB AICTE funded project.

Kompelly Sreeja, Kompelly Sreeja, student pursuing final year of Computer Science and Engineering at Gokaraju Rangaraju Institute of Engineering and Technology. Her interesting domain is soft computing. Her projects in the domain of soft computing include Disease prediction based on symptoms, Offline signature verification. She has good knowledge of Python and completed the projects using Python language.

Sai Anoushka K, Sai Anoushka K, is currently pursuing the final year of Computer Science and Engineering at Gokaraju Rangaraju Insitute of Engineering and Technology. She pursued her schooling at Meridian School, Banjara Hills. Her interest lies in Artificial Intelligence. She finished a project in the same domain that help detect counterfeit currency from real currency and is currently working on offline signature verification. She has good knowledge of Python and C language and has completed a couple of projects in the same.

Telugu Akila, Telugu Akila, is a hardworking engineering student pursuing the fourth year of B.Tech at Gokaraju Rangaraju Institute of Engineering and Technology specialising in Computer Science and Engineering with an overall CGPA of 9.4. She has a convincing knowledge of Python and Machine Learning. She had worked on projects which include Predicting the Diseases Based on Symptoms which comes under the domain of Soft Computing which has a major portion of machine learning concepts and python programming, she also had a great contribution to the completion of the project- Offline Signature Verification.

Sneha Nimmala, Sneha Nimmala, is currently in the final year of Computer Science and Engineering at Gokaraju Rangaraju Institute of Engineering and Technology. Her interest lies in Data Structures. She worked on a project in the domain of Artificial Intelligence and Machine Learning where she trained a model to detect fake currency from the real. She has a good grasp of Python and Data Structures.