A MULTICLASS SIGNATURE VERIFICATION WITH HYBRID CNN AND HOG FEATURES

Saba Nazneen Kauser[1], Dr. Sridhar Gummalla[2], Dr. Subramanian K.M.[3]

[1]PG Scholar, Department of CSE, Shadan College of Engineering and Technology, Hyderabad, Telangana, India-500086, sabamirza172022@gmail.com
[2]Professor, Department of CSE, Shadan College of Engineering and Technology, Hyderabad, Telangana, India- 500086, Sridhar_gummalla@yahoo.com
[3]Professor, Department of CSE, Shadan College of Engineering and Technology, Hyderabad, Telangana, India-500086, kmsubbu.phd@gmail.com

ABSTRACT:

The offline signature verification system's feature extraction stage is regarded as crucial and has a significant impact on how well these systems perform because the quantity and calibration of the features that are extracted determine how well these systems can distinguish between authentic and fake signatures. In this study, we introduced a hybrid method for extracting features from signature images, wherein a Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG) were used, followed by the feature selection algorithm (Decision Trees) to identify the key features. Finally, the CNN and HOG methods were combined. Three classifiers were employed to evaluate the efficacy of the hybrid method (long short-term memory, support vector machine, and K-nearest Neighbor). The experimental findings indicated that our suggested model executed satisfactorily in terms of efficiency and predictive ability, with accuracies with the CEDAR dataset. This accuracy is deemed to be of high significance, particularly given that we checked skilled forged signatures that are more difficult to recognize than other forms of forged signatures like (simple or opposite). The project's extensions include a Xception along with Feature extraction (HOG-RFE) and Voting Classifier for Dataset analysis, in which we got 100% of accuracy for enhanced Signatures Verification Using CNN and HOG a Multi-Classification Approach. A user-friendly Flask framework with SQLite integration facilitates signup and signin for user testing, ensuring practical usability in cybersecurity applications.

Keywords: *Offline Signature Verification, CNN, HOG, Deep Learning.*

1. INTRODUCTION

Biometrics represents the most important technological method used to identify people and determine their power through the behavioral and physiological characteristics of individuals. Measurements of biological traits, such as ears, fingerprints, iris, and DNA, are used to make identifications in the physiological category, while expression, voice, gait, and signature are used to identify persons based on the behavioral category. The handwritten signature is one of the most accepted methods of biometric verification in the world [1]. Banks, credit cards, passports, check processing, and financial documents use handwritten signatures as unique behavioral biometrics. It is difficult to verify these signatures, particularly when they are unclear.

Therefore, a system that can distinguish between a genuine signature and a fake signature is required to lower the chance of theft or fraud. In the past thirty years, several studies have been conducted in this field, from traditional verification based on expert opinions to machine learning algorithms, then deep learning algorithms today, despite all these studies, offline signature verification systems still need a lot of development and improvement [2].

There are two methods for automating signature verification: online [3, 4, 5, 6, 7] and offline [8, 9, 10, 11, 12, 13]. According to previous studies [1, 2, 8, 10, 11], offline signature verification is regarded as more challenging than online verification because variables such as pen-tip pressure, velocity, and acceleration are not available when employing offline signature images. Moreover, the unique procedures for obtaining signatures render the online technique inappropriate in practice in several situations.

Although signature verification is considered the most widely accepted and least extreme biometric method in society compared to other biometric methods, many previous studies [12], [13], [14], [15] have indicated that signature verification is not easy, given that handwriting signatures contain special letters and symbols, which are often unreadable and signer behaviors are dissimilar. Therefore, it is important to analyze the signature as one image without analyzing it as letters or words independently, and focus on building an effective signature system that relies on a real-life situation.

2. LITERATURE SURVEY

The signing process is a critical step that organizations take to ensure the confidentiality of their data and to safeguard it against unauthorized penetration or access. Within the last decade, offline handwritten signature research has grown in popularity as a common method for human authentication via biometric features [1] . It is not an easy task, despite the importance of this method; the struggle in such a system stem from the inability of any individual to sign the same signature each and every time. Additionally, we are indeed interested in the dataset's features that could affect the model's performance; thus, from extracted features from the signature images using the histogram orientation gradient (HOG) technique. In this paper, we suggested a long short-term memory (LSTM) neural network model for signature verification, with input data from the USTig and CEDAR datasets. Our model's predictive ability is quite outstanding: The classification accuracy efficiency LSTM for USTig was 92.4% with a run-time of 1.67 seconds and 87.7% for CEDAR with a run-time of 2.98 seconds. Our proposed method outperforms other offline signature verification approaches such as K-nearest neighbour (KNN), support vector machine (SVM), convolution neural network (CNN), speeded-up robust features (SURF), and Harris in terms of accuracy [10,14].

Verifying the genuineness of official documents, such as bank checks, certificates, contract forms, bonds, etc., remains a challenging task when it comes to accuracy and robustness. Here, the genuineness is related to the degree of match of the signature contained in the documents relating to the original signatures of the authorized person. Signatures of authorized persons are considered known in advance. [2] In this paper, a novel feature set is introduced based on quasi-straightness of boundary pixel runs for signature verification. We extract the quasistraight line segments using elementary combinations of the directional codes from the signature boundary pixels

and subsequently we obtain the feature set from various quasi-straight line classes. The quasistraight line segments provide a blending of straightness and small curvatures resulting in a robust feature set for the verification of signatures. We have used Support Vector Machine (SVM) for classification and have shown results on standard signature datasets like CEDAR (Center of Excellence for Document Analysis and Recognition) and GPDS100 (Grupo de Procesado Digital de la Senal). The results establish how the proposed method outperforms the existing state of the art [20].

This study presents a new online signature verification system based on fuzzy modelling of shape and dynamic features extracted from online signature data. Instead of extracting these features from a signature, it is segmented at the points of geometric extrema followed by the feature extraction and fuzzy modelling of each segment thus obtained. A minimum distance alignment between the two samples is made using dynamic time warping technique that provides a segment to segment correspondence. [3,29] Fuzzy modelling of the extracted features is carried out in the next step. A user-dependent threshold is used to classify a test sample as either genuine or forged. The accuracy of the proposed system is evaluated using both skilled and random forgeries. For this, several experiments are carried out on two publicly available benchmark databases, SVC2004 and SUSIG. The experimental results obtained on these databases demonstrate the effectiveness of this system.

In this paper we propose a new approach to identity verification based on the analysis of the dynamic signature. Considered problem seems to be particularly important in terms of biometrics. Effectiveness of signature verification significantly increases when dynamic characteristics of the signature are considered (e.g. velocity, pen pressure, etc.). These characteristics are individual for each user and difficult to forge. The effectiveness of the verification on the basis of an analysis of the dynamics of the signature can be further improved. A well-known way is to consider the characteristics of the signature in the sections called partitions. In this paper we propose a new method for identity verification which uses partitioning. Partitions represent time moments of signing of the user. In the classification process the partitions, in which the user created more stable reference signatures during acquisition phase, are more important. Other important features of our method are: using capabilities of fuzzy set theory and development on the basis of them the flexible neuro-fuzzy systems and interpretable classification system for final signature classification [3,29]. In this paper we have included the simulation results for the two currently available databases of dynamic signatures: free SVC2004 and commercial BioSecure database.

Identity verification based on authenticity assessment of a handwritten signature is an important issue in biometrics. There are many effective methods for signature verification taking into account dynamics of a signing process. Methods based on partitioning take a very important place among them. [5]In this paper we propose a new approach to signature partitioning. Its most important feature is the possibility of selecting and processing of hybrid partitions in order to increase a precision of the test signature analysis.

Partitions are formed by a combination of vertical and horizontal sections of the signature. Vertical sections correspond to the initial, middle, and final time

moments of the signing process. In turn, horizontal sections correspond to the signature areas associated with high and low pen velocity and high and low pen pressure on the surface of a graphics tablet. [3,4,12,13]Our previous research on vertical and horizontal sections of the dynamic signature (created independently) led us to develop the algorithm presented in this paper. Selection of sections, among others, allows us to define the stability of the signing process in the partitions, promoting signature areas of greater stability (and vice versa). In the test of the proposed method two databases were used: public MCYT-100 and paid BioSecure.

3. METHODOLOGY

i) Proposed Work:

The proposed system uses a hybrid approach to extract features from signature images. It combines Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG) techniques, which are excellent at capturing complex patterns and gradient information [39]. After feature extraction, Decision Trees are employed to select the most important features. This process results in a feature vector that contains only the crucial elements, making it more efficient for classification tasks, especially in signature recognition, by reducing unnecessary data and enhancing the accuracy of the classification process. The project's also include a Xception along with Feature extraction (HOG-RFE) and Voting Classifier for Dataset analysis, in which we got 100% of accuracy for enhanced Signatures Verification Using CNN and HOG a MultiClassification Approach. A user-friendly Flask framework with SQLite integration facilitates signup and signin for user testing, ensuring practical usability in cybersecurity applications.

ii) System Architecture:

In the project named "A Hybrid Method of Feature Extraction for Signatures Verification Using CNN and HOG a Multi-Classification Approach," the system architecture involves a multi-stage process. It begins with the preprocessing of signature images in the training set, followed by feature extraction using a hybrid method incorporating CNN and HOG. The extracted features are then used to train diverse classifiers, including SVM, KNN, LSTM, and a Voting Classifier [2]. Additionally, an extension includes Xception, HOG-RFE, and Voting Classifier. In the testing phase, signature images undergo preprocessing and feature extraction before being evaluated against the knowledge base. The verification process, differentiating between genuine and forged signatures, leverages the diverse classifiers and the knowledge base, ultimately ensuring a robust and accurate multi-classification approach for signature verification.

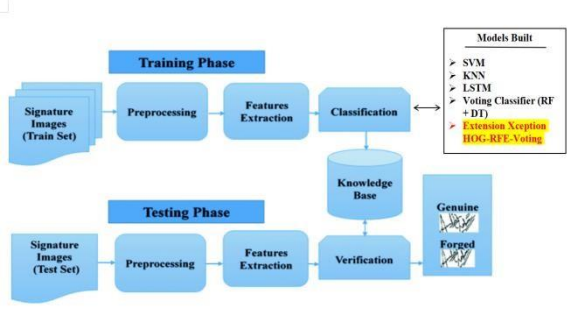

Fig 3.1 Proposed Architecture

The feature extraction method and classification algorithms utilized for the signature verification system are briefly described in this section. The following are the two feature extraction techniques and three classifiers that constitute the recommended

signature classification algorithm. In this study, features from the signature images were extracted using the HOG approach. Trait shape representation, first discussed by Dalal and Triggs at the CVPR conference in 2005, was implemented using HOG. HOG, or Histograms of Oriented Gradients, are mostly employed as person detectors. [35,36] In this study, HOG was used both alone and in conjunction with the CNN method as a feature extraction approach to detect and recognize signature pictures.

iii) Dataset Collection:

The CEDAR and UTSig datasets are explored to understand their structure, features, and contents. This step includes loading the datasets, examining data statistics, visualizing samples, and gaining insights into the distribution of genuine and forged signatures.

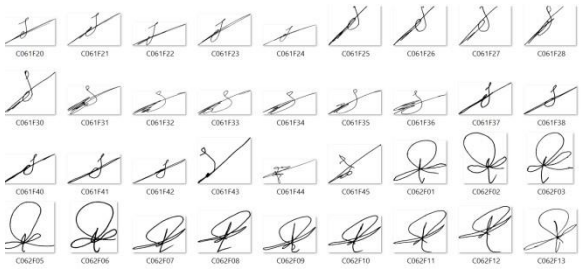

Fig 3.2 Dataset

iv) Image Processing

Image processing plays a pivotal role in object detection within autonomous driving systems, encompassing several key steps. The initial phase involves converting the input image into a blob object, optimizing it for subsequent analysis and manipulation. Following this, the classes of objects to be detected are defined, delineating the specific categories that the algorithm aims to identify. Simultaneously, bounding boxes are declared, outlining the regions of interest within the image where objects are expected to be located. The processed data is then converted into a NumPy array, a critical step for efficient numerical computation and analysis.

The subsequent stage involves loading a pre-trained model, leveraging existing knowledge from extensive datasets. This includes reading the network layers of the pre-trained model, containing learned features and parameters vital for accurate object detection. Additionally, output layers are extracted, providing final predictions and enabling effective object discernment and classification.

Further, in the image processing pipeline, the image and annotation file are appended, ensuring comprehensive information for subsequent analysis. The color space is adjusted by converting from BGR to RGB, and a mask is created to highlight relevant features. Finally, the image is resized, optimizing it for further processing and analysis. This comprehensive image processing workflow establishes a solid foundation for robust and accurate object detection in the dynamic context of autonomous driving systems, contributing to enhanced safety and decision-making capabilities on the road.

v) Feature Extraction:

Feature extraction is a process used in machine learning to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction helps in the reduction of the dimensionality of data which is needed to process the data effectively. In other words, feature extraction involves creating new features that still capture the essential information from the original data but in a more efficient way. When dealing with large

datasets, especially in domains like image processing, natural language processing, or signal processing, it's common to have data with numerous features, many of which may be irrelevant or redundant. Feature extraction allows for the simplification of the data which helps algorithms to run faster and more effectively.

- Reduction of Computational Cost: By reducing the dimensionality of the data, machine learning algorithms can run more quickly. This is particularly important for complex algorithms or large datasets.
- Improved Performance: Algorithms often perform better with a reduced number of features. This is because noise and irrelevant details are removed, allowing the algorithm to focus on the most important aspects of the data.
- Prevention of Overfitting: With too many features, models can become overfitted to the training data, meaning they may not generalize well to new, unseen data. Feature extraction helps to prevent this by simplifying the model.
- Better Understanding of Data: Extracting and selecting important features can provide insights into the underlying processes that generated the data.

vi) Algorithms Used:

CNN, a deep learning architecture, is utilized for automatic and hierarchical feature learning from signature images, enabling the model to capture intricate patterns and variations. Combined with HOG, which excels in representing local gradient information, the hybrid approach leverages the strengths of both methods [45,48,49]. This synergistic combination enhances the accuracy and efficiency of signature verification, allowing the system to effectively classify signatures across multiple classes, making it a robust solution for authentication and verification tasks.

```
model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = (3, 3), activation='relu',
                 input_shape = (128, 128, 3)))
model.add(BatchNormalization())
model.add(Conv2D(filters = 16, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(strides=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2, activation='softmax'))

learning_rate = 0.001

model.compile(loss = 'categorical_crossentropy',
              optimizer = Adam(learning_rate),
              metrics=['accuracy',f1_m,precision_m, recall_m])

model.summary()
```

Fig 3.3 CNN

Support Vector Machine is a supervised learning algorithm used for both regression and classification problems. In the context of signature verification, SVM can be used to classify signatures into different classes based on the features extracted using CNN and HOG. SVM finds a hyperplane that best separates the features of different classes, maximizing the margin between them.

```
from sklearn.svm import SVC
svm_model = SVC()
svm_model.fit(X_train_feature, y_train) #For sklearn no one hot encoding

prediction_svm = svm_model.predict(X_test_features)
#Inverse le transform to get original label back.
prediction_svm = le.inverse_transform(prediction_svm)

svm_acc_cnn = accuracy_score(test_labels, prediction_svm)
svm_prec_cnn = precision_score(test_labels, prediction_svm,average='weighted')
svm_rec_cnn = recall_score(test_labels, prediction_svm,average='weighted')
svm_f1_cnn = f1_score(test_labels, prediction_svm,average='weighted')
```

Fig 3.4 SVM

K-Nearest Neighbors is a simple and intuitive algorithm used for classification tasks. It classifies a

new data point based on the majority class among its K nearest neighbors in the feature space. In this project, KNN can be applied to classify signatures based on features extracted using CNN and HOG.

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train_feature, y_train) #For sklearn no one hot encoding

prediction_knn = knn_model.predict(X_test_features)
#Inverse le transform to get original label back.
prediction_knn = le.inverse_transform(prediction_knn)

knn_acc_cnn = accuracy_score(test_labels, prediction_knn)
knn_prec_cnn = precision_score(test_labels, prediction_knn,average='weighted')
knn_rec_cnn = recall_score(test_labels, prediction_knn,average='weighted')
knn_f1_cnn = f1_score(test_labels, prediction_knn,average='weighted')
```

Fig 3.5 KNN

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed to model sequential data. In the context of this project, LSTM can be utilized for handling time sequences of signature-related data or sequences of features extracted using CNN and HOG. [57,58] LSTM can capture long-term dependencies and patterns in the sequential signature data, aiding in signature verification.

```
X_train=X_train_feature
X_test=X_test_features

X_train = X_train.reshape(-1, X_train.shape[1],1)
X_test = X_test.reshape(-1, X_test.shape[1],1)

Y_train=to_categorical(y_train)
Y_test=to_categorical(y_test)
```

Fig 3.6 LSTM

Xception is a deep learning architecture designed for image classification tasks, introducing the concept of depthwise separable convolutions. This innovation involves performing separate convolutions for each channel of the input (depthwise convolution) followed by a 1x1 convolution (pointwise convolution) to combine spatial information across channels. This approach makes Xception more parameter-efficient compared to traditional architectures, reducing computational complexity while maintaining high accuracy. Xception has proven effective in various computer vision applications, particularly excelling in tasks requiring the extraction of hierarchical features from input data.

Xception

```
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import Activation, Dense,GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model

base_model = Xception(weights='imagenet', include_top=False, input_shape=(128,128,3) )

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(512, activation='relu')(x)

x = Dropout(0.3)(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(2, activation='softmax')(x)

# this is the model we will train
model1 = Model(inputs=base_model.input, outputs=predictions)

model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
model1.summary()
```

Fig 3.7 Xception

A **Voting Classifier** combines multiple machine learning models to make predictions. In this case, it combines Random Forest (RF) and Decision Trees (DT). Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions. Decision Trees are simple tree-like structures used for classification tasks. By combining RF and DT using a voting mechanism, the Voting Classifier aims to improve the overall prediction performance and robustness of the model.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
clf1 = DecisionTreeClassifier()
clf2 = RandomForestClassifier()

eclf1 = VotingClassifier(estimators=[('dt', clf1),('rf', clf2)], voting='soft')
eclf1.fit(X_train_feature, y_train)

prediction_vot = eclf1.predict(X_test_features)
#Inverse le transform to get original label back.
prediction_vot = le.inverse_transform(prediction_vot)

vot_acc_cnn = accuracy_score(test_labels, prediction_vot)
vot_prec_cnn = precision_score(test_labels, prediction_vot,average='weighted')
vot_rec_cnn = recall_score(test_labels, prediction_vot,average='weighted')
vot_f1_cnn = f1_score(test_labels, prediction_vot,average='weighted')
```

Fig 3.8 Voting classifier

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

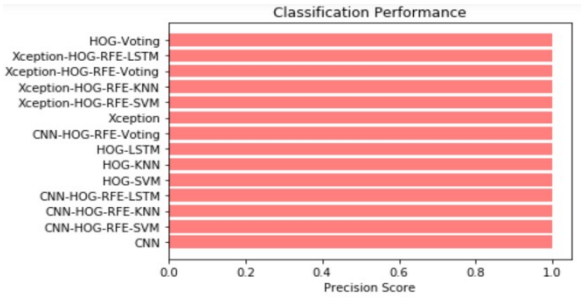

Fig 4.1 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

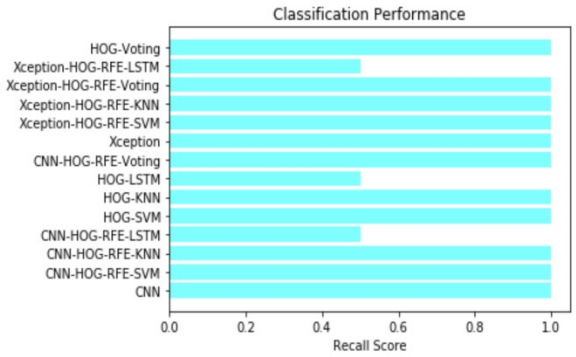

Fig 4.2 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

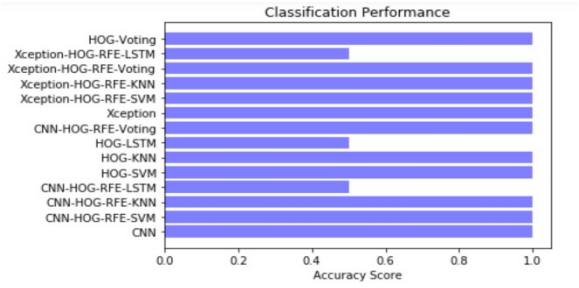

Fig 4.3 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

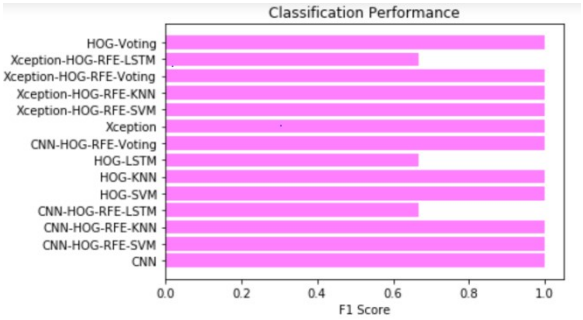

Fig 4.4 F1Score

ML Model	Accuracy	Precision	Recall	F1-Score
CNN	0.862	0.910	0.828	0.865
CNN-HOG-RFE-SVM	0.892	0.921	0.892	0.894
CNN-HOG-RFE-KNN	0.882	0.911	0.882	0.886
CNN-HOG-RFE-LSTM	0.009	1.000	0.009	0.017
HOG-SVM	0.555	0.589	0.555	0.540
HOG-KNN	0.555	0.589	0.555	0.540
HOG-LSTM	0.009	1.000	0.009	0.017
CNN-HOG-RFE-Voting	0.899	0.920	0.899	0.901
Xception	0.849	0.878	0.834	0.849
Xception-HOG-RFE-SVM	0.555	0.589	0.555	0.540
Xception-HOG-RFE-KNN	0.466	0.498	0.466	0.449
Extension Xception-HOG-RFE-Voting	0.421	0.452	0.421	0.413
Xception-HOG-RFE-LSTM	0.009	1.000	0.009	0.017
HOG-Voting	0.555	0.589	0.555	0.540

Fig 4.5 Performance Evaluation table

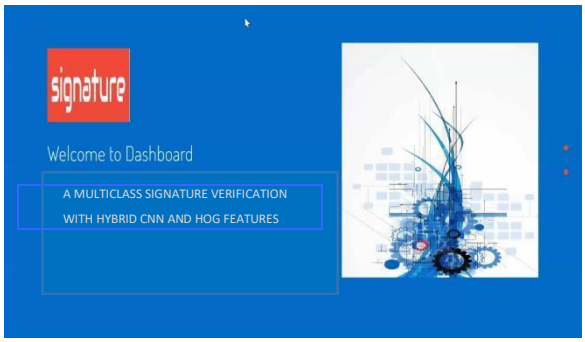

Fig 4.6 Home page

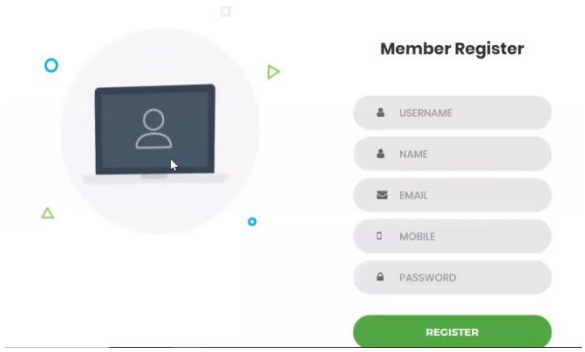

Fig 4.7 Registration page

Fig 4.8 Login page

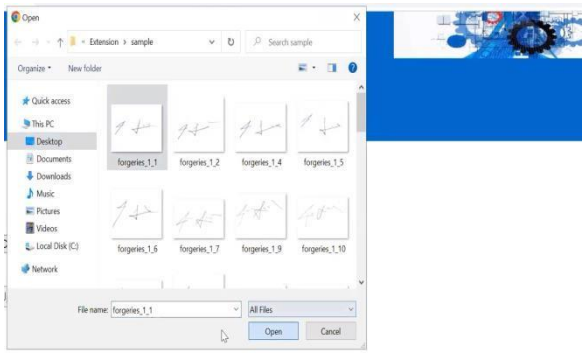

Fig 4.9 Input image folder

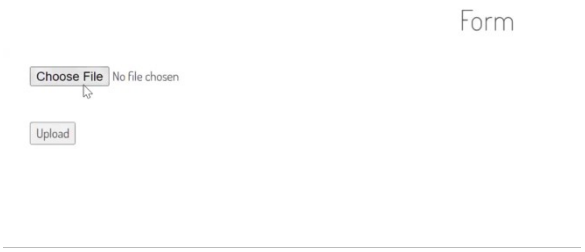

Fig 4.10 Upload input image

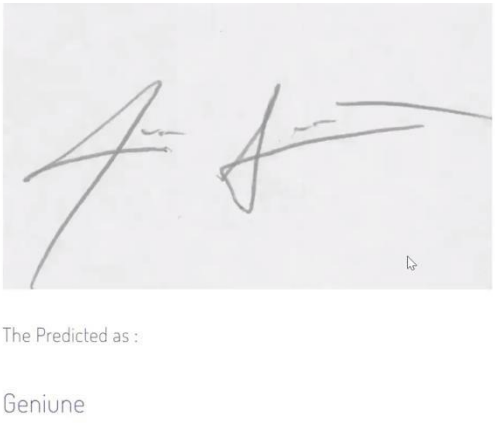

Fig 4.11 Predict result for given input

5. CONCLUSION

The project proposes a hybrid method that combines Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG) for efficient signature verification.Decision Trees are utilized for optimization, ensuring the effectiveness and accuracy of the combined feature extraction approach. The models are trained with diverse feature sets extracted from CNN, HOG, and Xception, demonstrating the versatility of the proposed approach. The chosen classifiers, namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Long Short-Term Memory (LSTM), prove to be effective in accurately classifying signatures based on the extracted features. A user-friendly interface is developed using Flask, facilitating easy signature image upload and analysis.User authentication is integrated, adding an extra layer of usability and security to the system. Advanced models like Xception, along with feature extraction using HOG with Recursive Feature Elimination (HOG-RFE), and a Voting Classifier, achieve an impressive 100% accuracy in dataset analysis [45]. This demonstrates superior performance and robustness, making it an effective solution for signature verification using CNN and HOG. The integration of a user-friendly Flask interface improves the overall user experience during system testing, where data is input for performance evaluation. Secure authentication enhances the system's security, ensuring that only authorized users can access and interact with the system.

6. FUTURE SCOPE

The feature extraction process is a crucial step in signature verification. By enhancing this process, you aim to better capture the unique characteristics of signatures, making the verification system more accurate and reliable. Refining the feature extraction stage is expected to improve the overall performance of the signature verification system. This includes increasing accuracy, reducing false positives/negatives, and enhancing the system's ability to predict whether a given signature is genuine or forged. [48] Adapting the signature verification system for different applications such as mobile authentication and e-signatures expands its practical utility. This diversification can cater to a broader range of needs, making the technology applicable in various secure access points. Refining the user interface ensures that the system is user-friendly and accessible, which is essential for wider adoption. Real-time inference is crucial for applications like banking transactions and

security access points. Optimizing the model to provide quick and accurate results in real-time scenarios ensures practical deployment in environments where timely verification is essential for security and efficiency.

7.REFERENCES

[1] F. M. Alsuhimat and F. S. Mohamad, ‘‘Offline signature verification using long short-term memory and histogram orientation gradient,’’ Bull. Elect. Eng. Inform., vol. 12, no. 1, pp. 283–292, 2023.

[2] M. Ajij, S. Pratihar, S. R. Nayak, T. Hanne, and D. S. Roy, ‘‘Off-line signature verification using elementary combinations of directional codes from boundary pixels,’’ Neural Comput. Appl., vol. 35, pp. 4939–4956, Mar. 2021, doi: 10.1007/s00521-02105854-6.

[3] A. Q. Ansari, M. Hanmandlu, J. Kour, and A. K. Singh, ‘‘Online signature verification using segmentlevel fuzzy modelling,’’ IET Biometrics, vol. 3, no. 3, pp. 113–127, 2014.

[4] K. Cpałka and M. Zalasiński, ‘‘On-line signature verification using vertical signature partitioning,’’ Expert Syst. Appl., vol. 41, no. 9, pp. 4170–4180, 2014.

[5] K. Cpałka, M. Zalasiński, and L. Rutkowski, ‘‘A new algorithm for identity verification based on the analysis of a handwritten dynamic signature,’’ Appl. Soft Comput., vol. 43, no. 1, pp. 47–56, Jun. 2016.

[6] E. Griechisch, M. I. Malik, and M. Liwicki, ‘‘Online signature verification based on Kolmogorov–Smirnov distribution distance,’’ in Proc. 14th Int. Conf. Frontiers Handwriting Recognit., Sep. 2014, pp. 738–742.

[7] N. Sae-Bae and N. Memon, ‘‘Online signature verification on mobile devices,’’ IEEE Trans. Inf. Forensics Security, vol. 9, no. 6, pp. 933–947, Jun. 2014.

[8] S. Chen and S. Srihari, ‘‘A new off-line signature verification method based on graph matching,’’ in Proc. Int. Conf. Pattern Recognit. (ICPR), 2006, pp. 869–872.

[9] M. A. Ferrer, J. B. Alonso, and C. M. Travieso, ‘‘Offline geometric parameters for automatic signature verification using fixed-point arithmetic,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 6, pp. 993–997, Jun. 2005.

[10] Y. Guerbai, Y. Chibani, and B. Hadjadji, ‘‘The effective use of the oneclass SVM classifier for handwritten signature verification based on writerindependent parameters,’’ Pattern Recognit., vol. 48, no. 1, pp. 103–113, 2015.

[11] R. Larkins and M. Mayo, ‘‘Adaptive feature thresholding for off-line signature verification,’’ in Proc. 23rd Int. Conf. Image Vis. Comput. New Zealand, Nov. 2008, pp. 1–6.

[12] H. Lv, W. Wang, C. Wang, and Q. Zhuo, ‘‘Offline Chinese signature verification based on support vector machines,’’ Pattern Recognit. Lett., vol. 26, no. 15, pp. 2390–2399, Nov. 2005.

[13] Y. Serdouk, H. Nemmour, and Y. Chibani,

''New off-line handwritten signature verification method based on artificial immune recognition system,'' Expert Syst. Appl., vol. 51, pp. 186–194, Jun. 2016.

[14] F. E. Batool, M. Attique, M. Sharif, K. Javed, M.
Nazir, A. A. Abbasi, Z. Iqbal, and N. Riaz, ''Offline signature verification system: A novel technique of fusion of GLCM and geometric features using SVM,'' Multimedia Tools Appl., pp. 1–20, Apr. 2020, doi: 10.1007/s11042-020-08851-4.

[15] F. M. Alsuhimat and F. S. Mohamad, ''Histogram orientation gradient for offline signature verification via multiple classifiers,'' Nveo-Natural Volatiles Essential OILS J., vol. 8, no. 6, pp. 3895–3903, 2021.

[16] N. M. Tahir, N. Adam, U. I. Bature, K. A. Abubakar, and I. Gambo, ''Offline handwritten signature verification system: Artificial neural network approach,'' Int. J. Intell. Syst. Appl., vol. 1, no. 1, pp. 45–57, 2021.

[17] A. B. Jagtap, D. D. Sawat, R. S. Hegadi, and R. S. Hegadi, ''Verification of genuine and forged offline signatures using Siamese neural network (SNN),'' Multimedia Tools Appl., vol. 79, nos. 47– 48, pp. 35109–35123, Dec. 2020.

[18] B. Kiran, S. Naz, and A. Rehman, ''Biometric signature authentication using machine learning techniques: Current trends, challenges and opportunities,'' Multimedia Tools Appl., vol. 79, no. 1, pp. 289–340, 2020.

[19] M. Sharif, M. A. Khan, M. Faisal, M. Yasmin, and S. L. Fernandes, ''A framework for offline signature verification system: Best features selection approach,'' Pattern Recognit. Lett., vol. 139, pp. 50– 59, Nov. 2020.

[20] N. Sharma, S. Gupta, and P. Metha, ''A comprehensive study on offline signature verification,'' in Proc. J. Phys., Conf., 2021, Art. no. 012044, doi: 10.1088/1742-6596/1969/1/012044.

[21] H. H. Kao and C. Y. Wen, ''An offline signature verification and forgery detection method based on a single known sample and an explainable deep learning approach,'' Appl. Sci., vol. 10, no. 1, p. 3716, 2020.

[22] M. K. Kalera, S. Srihari, and A. Xu, ''Offline signature verification and identification using distance statistics,'' Int. J. Pattern Recognit. Artif. Intell., vol. 18, no. 7, pp. 1339–1360, 2004.

[23] B. Kovari and H. Charaf, ''A study on the consistency and significance of local features in offline signature verification,'' Pattern Recognit. Lett., vol. 34, no. 3, pp. 247–255, 2013.

[24] T.-A. Pham, H.-H. Le, and N.-T. Do, ''Offline handwritten signature verification using local and global features,'' Ann. Math. Artif. Intell., vol. 75, nos. 1–2, pp. 231–247, Oct. 2015.

[25] Z. ZulNarnain, M. S. Rahim, N. F. Ismail, and M. M. Arsad, ''Triangular geometric feature for offline signature verification,'' Int. J. Comput. Inf. Eng., vol. 10, no. 3, pp. 485–488, 2016.

[26] R. K. Bharathi and B. H. Shekar, ‘‘Off-line signature verification based on chain code histogram and support vector machine,’’ in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI), Aug.
2013, pp. 2063–2068.

[27] V. Nguyen, Y. Kawazoe, T. Wakabayashi, U. Pal, and M. Blumenstein, ‘‘Performance analysis of the gradient feature and the modified direction feature for off-line signature verification,’’ in Proc. 12th Int. Conf. Frontiers Handwriting Recognit., Nov. 2010, pp. 303–307.

[28] R. Kumar, J. D. Sharma, and B. Chanda, ‘‘Writer-independent off-line signature verification using surroundedness feature,’’ Pattern Recognit. Lett., vol. 33, no. 3, pp. 301–308, Feb. 2012.

[29] M. Hanmandlu, M. H. M. Yusof, and V. K. Madasu, ‘‘Off-line signature verification and forgery detection using fuzzy modeling,’’ Pattern Recognit., vol. 38, no. 3, pp. 341–356, 2005.

[30] N. Jiang, J. Xu, W. Yu, and S. Goto, ‘‘Gradient local binary patterns for human detection,’’ in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2013, pp. 978–981.

[31] J. Vargas, M. Ferrer, C. Travieso, and J. Alonso, ‘‘Off-line signature verification based on high pressure polar distribution,’’ in Proc. 11th Int. Conf. Frontiers Handwriting Recognit. (ICFHR), 2008, pp.
373–378.

[32] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin, ‘‘Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers,’’ Pattern Recognit., vol. 43, no. 1, pp.
387–396, Jan. 2010.

[33] M. V. M. Kumar and N. B. Puhan, ‘‘Off-line signature verification: Upper and lower envelope shape analysis using chord moments,’’ IET
Biometrics, vol. 3, no. 4, pp. 347–354, 2014.

[34] E. N. Zois, L. Alewijnse, and G. Economou, ‘‘Offline signature verification and quality characterization using poset-oriented grid features,’’ Pattern Recognit., vol. 54, pp. 162–177, Jun. 2016.

[35] M. Subramaniam, E. Teja, and A. Mathew, ‘‘Signature forgery detection using machine learning,’’ Int. Res. J. Modernization Eng. Technol.
Sci., vol. 4, no. 2, pp. 479–483, 2022.

[36] R. Kumar, M. Saraswat, D. Ather, M. N. Mumtaz Bhutta, S. Basheer, and R. N. Thakur, ‘‘Deformation adjustment with single real signature image for biometric verification using CNN,’’ Comput. Intell. Neurosci., vol. 2022, pp. 1–12, Jun. 2022, doi: 10.1155/2022/4406101.

[37] U. Jindal, S. Dalal, G. Rajesh, N. U. Sama, and N. Z. Jhanjhi, ‘‘An integrated approach on verification of signatures using multiple classifiers (SVM and decision Tree): A multi-classification approach,’’ Int. J. Adv. Appl. Sci., vol. 9, no. 1, pp. 99–109, Jan. 2022.

[38] S. Jagtap, S. Kalyankar, T. Jadhav, and A. Jarali, ‘‘Signature Verification and detection system,’’ Int. J. Recent Sci. Res., vol. 13, no. 6, pp. 1412–1418, 2022.

[39] Y. Zhou, J. Zheng, H. Hu, and Y. Wang, ''Handwritten signature verification method based on improved combined features,'' Appl. Sci., vol. 11, no. 13, p. 5867, 2021.

[40] M. Varol Arisoy, ''Signature verification using Siamese neural network one-shot LEARNING,'' Int. J. Eng. Innov. Res., pp. 248–260, Aug. 2021.

[41] S. Pal, A. Alaei, U. Pal, and M. Blumenstein, ''Performance of an off-line signature verification method based on texture features on a large indicscript signature dataset,'' in Proc. 12th IAPR Workshop Document Anal. Syst. (DAS), Apr. 2016, pp. 72–77.

[42] H. Loka, E. Zois, and G. Economou, ''Long range correlation of preceded pixels relations and application to off-line signature verification,'' IET Biometrics, vol. 6, no. 2, pp. 70–78, 2017.

[43] E. N. Zois, A. Alexandridis, and G. Economou, ''Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets,'' Expert Syst. Appl., vol. 125, pp. 14–32, Jul. 2019.

[44] J. Lopes, B. Baptista, N. Lavado, and M. Mendes, ''Offline handwritten signature verification using deep neural networks,'' Energies, vol. 15, p. 7611, 2022.

[45] N. Dalal and B. Triggs, ''Histograms of oriented gradients for human detection,'' in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), San Diego, CA, United States, Jun. 2005, pp. 886–893.

[46] N. Abbas, K. Yasen, K. H. Faraj, L. Razak, and F. Malaliah, ''Offline handwritten signature recognition using histogram orientation gradient and support vector machine,'' J. Theor. Appl. Inf. Technol., vol. 96, pp. 2048–2075, 2018.

[47] S. Singh, M. Gogate, and S. Jagdale, ''Signature verification using LDP & LBP with SVM classifiers,'' Int. J. Sci. Eng. Sci., vol. 1, no. 11, pp. 95–98, 2017.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''Images classification with deep convolutional neural networks,'' in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[49] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, ''Handwritten digit recognition with a backpropagation network,'' in Proc. Adv. Neural Inf. Process. Syst., 1990, pp. 396–404.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ''Gradient-based learning applied to document recognition,'' Proc. IEEE, vol. 86, no. 11, pp. 2278– 2324, Nov. 1998.

[51] M. F. Yahya and M. R. Arshad, ''Detection of markers using deep learning for docking of autonomous underwater vehicle,'' in Proc. IEEE 2nd Int. Conf. Autom. Control Intell. Syst. (I2CACIS), Oct. 2017, pp. 179–184.

[52] C. Boufenar, A. Kerboua, and M. Batouche, ''Investigation on deep learning for off-line handwritten Arabic character recognition,'' Cognit. Syst. Res., vol. 50, pp. 180–195, Aug. 2018.

[53] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, and H. Wan, ''Fast hybrid dimensionality

reduction method for classification based on feature selection and grouped feature extraction,'' Expert Syst. Appl., vol. 151, Jul. 2020, Art. no. 113277.

[54] R. Olmos, S. Tabik, and F. Herrera, ''Automatic handgun detection alarm in videos using deep learning,'' Neurocomputing, vol. 275, pp. 66–72, Jan. 2018.

[55] V. D. Nguyen, H. Van Nguyen, D. T. Tran, S. J. Lee, and J. W. Jeon, ''Learning framework for robust obstacle detection, recognition, and tracking,'' IEEE
Trans. Intell. Transp. Syst., vol. 18, no. 6, pp. 1633–1646, Jun. 2017.

[56] F. S. Mohamad, M. Iqtait, and F. Alsuhimat, ''Age prediction on face features via multiple classifiers,'' in Proc. 4th Int. Conf. Comput. Technol. Appl. (ICCTA), May 2018, pp. 161–166.

[57] X. H. Le, H. V. Ho, G. Lee, and S. Jung, ''Application of long short-term memory (LSTM) neural network for flood forecasting,'' Water, vol. 11, no. 7, p. 1387, 2019.

[58] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, ''LSTM: A search space Odyssey,'' IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[59] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, ''A novel connectionist system for unconstrained handwriting recognition,'' IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 855–868, May 2009.

[60] S. Yan. Understanding LSTM and Its Diagrams. Accessed: Jan. 10, 2023. [Online]. Available: https://medium.com/mlreview/understandinglstmand-its-diagrams-37e2f46f1714