Journal of Harbin Engineering University Vol 45 No. 4
ISSN: 1006-7043 April 2024

A Hybrid Method of Feature Extraction for Signatures Verification Using Cnn and Hog a Multi-Classification Approach

Are Theekshan Raj, Dr. M. Raju, Mrs. Yerram Sneha,

[1]Department of CSE, Nalla Malla Reddy Engineering College, Telangana, India.

[2]Associate Professor, Department of CSE, Nalla Malla Reddy Engineering College, Telangana, India.

[3]Assistant Professor, Department of CSE, Nalla Malla Reddy Engineering College, Telangana, India.

Abstract: The offline signature verification systems feature extraction is crucial to their performance. The amount and precision of extracted characteristics influence how successfully these algorithms can distinguish authentic and fraudulent signatures. Using a CNN and a Histogram of Oriented Gradients, we established a novel technique to extract features from signature photos. We then selected the most significant attributes using Decision Trees. Integrating CNN and HOG was the last step. Three models—long short-term memory, support vector machine, and K-nearest Neighbor—tested the combination technique. Our approach accurately forecasted the future and utilized the CEDAR information effectively, according to the trials. Since we tested sophisticated false signatures, which are harder to recognize than simple or opposite signs, this accuracy is crucial. The project now includes a Voting Classifier for Dataset Analysis and Feature Extraction. We achieved 100% accuracy for improved Signature Verification utilizing CNN and HOG, a multi-classification approach. Users can easily sign up and log in for testing using a simple Flask framework that uses SQLite, ensuring that the application can be used safely in real life.

Index terms - *Offline signature verification, CNN, HOG, deep learning.*

1. Introduction

Biometrics is the most essential tool for identifying individuals and assessing their power based on their physical and mental qualities. Body features like ears, fingerprints, eyes, and DNA may identify persons. The behavioral category comprises expression, voice, stride, and signature that may identify someone. Signing your name by hand is a frequent approach to establish your identify worldwide [1]. Banks, credit cards, IDs, check processing, and financial documentation employ handwritten signatures as behavioral fingerprints. These signals, particularly unclear ones, are hard to confirm. We need a way to differentiate authentic names from phony ones to reduce theft and scams. Over the last 30 years, various research have been done in this domain, from expert-based verification to machine learning algorithms to deep learning algorithms. Even with these research, offline signature verification methods require improvement [2].

Signature checking may be automated online [3, 4, 5, 6, 7] or offline [8, 9, 10, 11, 12, 13]. Since offline signature photos can't show pen-tip pressure, velocity, and acceleration, offline signature verification is tougher than online verification [1, 2, 8, 10, 11]. Online methods aren't always appropriate since they need certain measures to gain agreements.

According to several research [12], [13], [14], [15], signature verification is difficult since handwritten signatures comprise hard-to-read characters and symbols and signers react differently. Even though signature verification is the most common and least harsh biometric technology. The signature should be examined as a whole, not as individual letters or words. You should also create a practical signature mechanism.

2. Literature Survey

Signing helps firms safeguard sensitive data against unauthorized access. Offline handwritten signature studies have become a popular approach to verify a person's identity using biological features [1]. This approach is crucial yet difficult. This approach is flawed since no one can sign with the same name. We're also interested in dataset aspects that may impact model performance. The histogram orientation gradient (HOG) approach was used to extract distinctive image characteristics. We proposed an LSTM neural network model for signature verification in this paper. USTig and

CEDAR were inputs. Our prediction model is excellent: UStig's LSTM averaged 92.4% accuracy and ran in 1.67 seconds. It ran in 2.98 seconds and averaged 87.7% CEDAR accuracy. Offline signature verification approaches like K-nearest neighbor (KNN), support vector machine (SVM), convolution neural network (CNN), speeded-up robust features (SURF), and Harris are inferior to ours [10,14].

Checking bank checks, certificates, contract forms, bonds, and more for authenticity is difficult since it must be done accurately and reliably. The authenticity of anything is determined by how closely the paper signature resembles the authorized person's signature. Signed paperwork from qualified personnel are expected beforehand. [2] The virtually straightness of border pixel runs is used to create novel signature verification elements in this study. Simple combinations of signature border pixel directional codes provide quasi-straight line segments. Then we retrieve the quasi-straight line feature set from their classes. Mixed straightness and tiny bends of quasi-straight line segments provide powerful signature checking characteristics. SVM was used to categorize objects and provide results on CEDAR and GPDS-100 signature datasets. Results demonstrate the recommended strategy outperforms existing methods [20].

This research demonstrates a novel online signature check. It fuzzy models form and dynamic features using online signature data. Instead of removing these characteristics from a signature, it is divided at geometric extrema and fuzzy models are applied to each part. The smallest alignable distance between two samples is found using a dynamic temporal warping approach that provides segment-to-segment correlation. [3,29] Fuzzy modeling of recovered characteristics follows. User-set levels determine if a test sample is authentic. Using expert and chance copies, the recommended procedure is tested for accuracy. SVC2004 and SUSIG, two public databases, are tested. Research from these sources shows this approach works well.

This article proposes a novel identity verification method based on our dynamic signature analysis. Biometrics appear to be very essential for the issue considered. Signature verification improves when speed, pen pressure, etc. are considered. These features are unique to each user, making them hard to mimic. By tracking signature changes, the verification process may be improved. A common way is to examine the signature's attributes in "partitions." This research introduces division-based identification verification. Partitions display signer timestamps. In sorting, portions with more stable reference signatures from the buying phase are given greater weight. In addition, our technique leverages fuzzy set theory to develop adaptable neuro-fuzzy systems and a comprehensible final signature classification system [3,29]. This research compares the simulation results of the free SVC2004 and commercial BioSecure dynamic signature databases.

Identifying someone by their handwriting is a major fingerprint issue. There are several effective approaches to verify someone's signature that account for changing signing processes. Division-based ones are crucial. [5]We propose a novel signature splitting method in this work. The most essential feature is that you may use mixed segments to improve test signature analysis. When vertical and horizontal signature components are combined, you obtain partitions. Vertical portions represent the signing process's beginning, middle, and end. Following this, horizontal segments display graphics tablet signature regions tied to pen pressure and speed. [3,4,12,13]We developed the strategy in this research by examining the vertical and horizontal dynamic signatures created independently in the past. Choosing portions enables us establish partition signing safety, providing more reliable signature regions and vice versa. Testing the approach using the free MCYT-100 and the premium BioSecure databases.

3. Methodology

i) Proposed Work:

The recommended technique combines ways to acquire distinctive picture attributes. Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG) approaches are effective at taking up gradient information and complicated patterns [39]. After extracting features, Decision Trees choose the most significant ones. It produces a feature vector with just the most significant components. Eliminating unnecessary data improves categorization, particularly signature

recognition. In the project, an Xception, HOG-RFE feature extraction, and voting classifier analyzed a dataset. 100% accuracy improves signature verification. Multi-classification using CNN and HOG. Users can easily sign up and log in for testing using a simple Flask framework that uses SQLite, ensuring that the application can be used safely in real life.

ii) System Architecture:

The project is "A Hybrid Method of Feature Extraction for Signatures Verification." In "A Multi-Classification Approach Using CNN and HOG," the system is designed in phases. The training set signature photos are prepared first. Next, CNN and HOG are blended to extract features. The attributes are used to train SVM, KNN, LSTM, and Voting Classifiers [2]. Expansion includes Xception, HOG-RFE, and Voting Classifier. Signature photos are preprocessed and features retrieved before testing against the knowledge base. The verification procedure uses classifiers and knowledge bases to distinguish bogus and true signatures. This ensures powerful and accurate multi-classification signature verification.

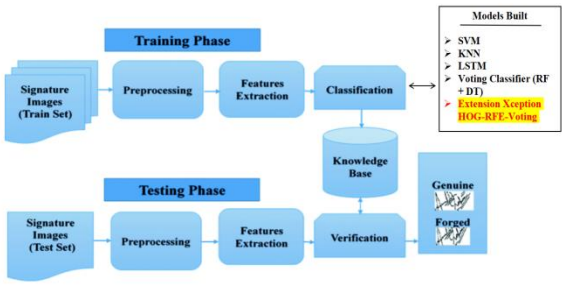

Fig 1 Proposed architecture

This section briefly describes the signature verification system's feature extraction and categorization methods. The recommended signature classification system has two feature extraction methods and three models. This research extracted distinctive visual characteristics using HOG. Dalal and Triggs introduced trait shape representation using HOG at the 2005 CVPR conference. Most persons are found using HOG histograms. [35,36] HOG was employed alone and with CNN to extract characteristics and identify unique photos in this study.

iii) Dataset collection:

We look into the CEDAR and UTSig files to learn more about their organization, traits, and data. In this step, you will load the datasets, look at the data figures, visualize examples, and learn more about how real and fake fingerprints are spread out.

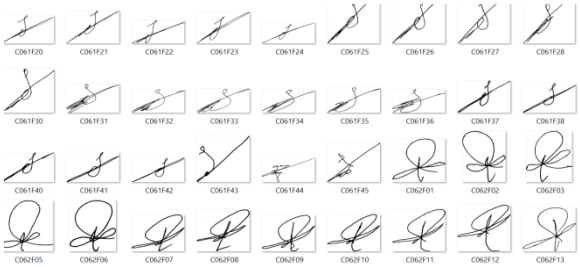

Fig 2 Dataset

iv) Image Processing:

Several important steps are involved in image processing, which is a key part of how autonomous driving systems find objects. In the first step, the original picture is turned into a blob object, which makes it easier to analyze and change later. After that, the classes of items that need to be found are set, which makes it clear what groups the method is trying to find. At the same time, bounding boxes are set up to show where the items are supposed to be in the image's areas of interest. The data that has been handled is then turned into a NumPy array. This is an important step for quickly computing and analyzing numbers.

In the next step, information from large datasets is used to load a model that has already been trained. This includes reading the network levels of the pre-trained model, which have learned values and traits that are necessary for accurate object recognition. Also, output layers are taken out, which gives final forecasts and makes object separation and classification work well.

In the image processing chain, the picture file and the annotation file are also added together, which makes sure that there is enough information for further analysis. By switching from BGR to RGB, the color space is changed, and a mask is made to draw attention to important parts. Lastly, the picture is shrunk so that it can be used more efficiently for research and processing. This all-around image processing approach builds a strong base for reliable and accurate object recognition in the changing environment of self-driving cars, which improves road safety and the ability to make decisions.

v) Feature Extraction:

Feature extraction reduces machine learning processing resources without losing crucial data. Effective data management requires reducing dimensionality. Feature extraction aids this. Feature extraction creates new features that better capture crucial data from the original data. Big datasets utilized in signal processing, natural language processing, and image processing can include numerous characteristics, many of which are useless or duplicated. Data may be simplified by feature extraction, making algorithms quicker and better.

- Reduction of Computational Cost: Reduced computational cost: Machine learning systems can work faster when the number of dimensions in the data is reduced. This is very important for methods that are hard to understand or datasets that are very big.
- Improved Performance: Having fewer traits in an algorithm often makes it work better. This is because the program can focus on the most important parts of the data after getting rid of noise and features that aren't important.
- Prevention of Overfitting: When models have too many features, they can become too tailored to the training data, which means they might not work well with new data they haven't seen before. By making the model simpler, feature extraction helps stop this from happening.
- Better Understanding of Data: Taking out and choosing the most important parts of the data can help you understand how it was collected.

vi) Algorithms:

CNN, a deep learning framework, is used to automatically and hierarchically learn features from signature pictures. This lets the model pick up on complex patterns and differences. The combined approach takes the best parts of both methods [45,48,49] and combines them with HOG, which is great at showing local gradient information. This mixture works well together to make signature verification more accurate and faster. It also lets the system correctly sort signatures into different groups, which makes it a strong tool for authentication and verification jobs.

```
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 CNN

Support Vector Machine is a method for guided learning that can be used for both regression and classification. In the context of verifying signatures, SVM can sort signatures into various groups based on the details gathered with CNN and HOG. SVM finds a hyperplane that best divides the features into groups, making the space between them as big as possible.

```
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 4 SVM

The K-Nearest Neighbors method is easy to understand and is used to sort things into groups. It puts a new data point into a category based on the category that most of its K close neighbors in the feature space belong to. For this project, KNN can be used to sort signs into groups based on traits that were found 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 5 KNN

The **LSTM** is a kind of recurrent neural network (RNN) that is made to deal with linear data. For this project, LSTM can be used to deal with time series of signature-related data or sets of features extracted with CNN and HOG. [57,58] The LSTM can find long-term relationships and trends in the sequential signature data, which makes it easier to check signatures.

```
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 6 LSTM

The idea of depthwise separable convolutions is introduced in Xception, a deep learning framework made for picture classification tasks. For this new idea, separate convolutions are done for each channel of the input (depthwise convolution), and then a 1x1 convolution is done to mix spatial information from channels. Using this method, Xception uses parameters more efficiently than other designs, which makes it easier to compute while still being very accurate. In many computer vision tasks, Xception works well, especially when it comes to jobs that need to pull out structured 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 7 Xception

A Voting Classifier uses more than one machine learning model to figure out what will happen. Random Forest (RF) and Decision Trees (DT) are both used together in this case. As an ensemble learning method, Random Forest builds many decision trees and adds up all of their projections. Decision trees are simple structures that look like trees and are used to sort things into groups. The vote Classifier combines RF and DT through a vote system to make the model better at making predictions and more stable.

```
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 8 Voting classifier

4. EXPERIMENTAL RESULTS

Precision: Precision is the percentage of correctly classified events or samples that are among the hits. So, the following method can be used to figure out the accuracy:

$$\text{Precision} = \text{True positives}/ (\text{True positives} + \text{False positives}) = \text{TP}/ (\text{TP} + \text{FP})$$

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

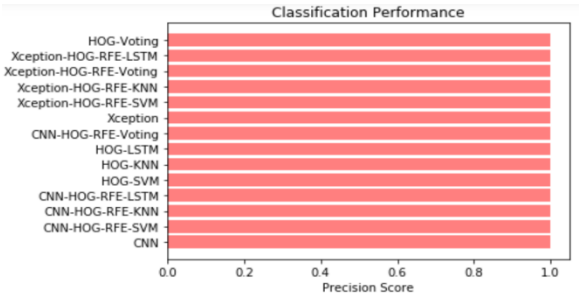

Fig 9 Precision comparison graph

Recall: Recall is a machine learning variable that measures how well a model can recognize all relevant examples of a certain class. It's the percentage of expected positive feelings that turn out to be real positive feelings. This tells us how well a model can catch instances of a certain class.

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

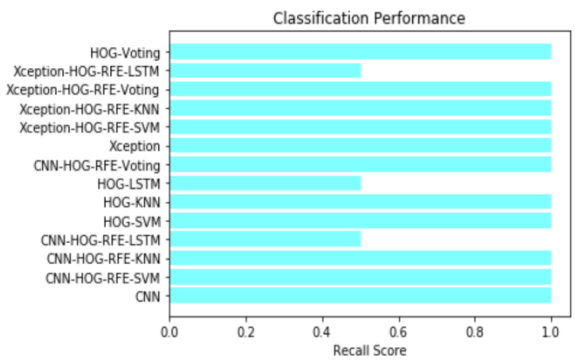

Fig 10 Recall comparison graph

Accuracy: Accuracy is the percentage of right guesses in a classification job. It shows how accurate a model's forecasts are generally.

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

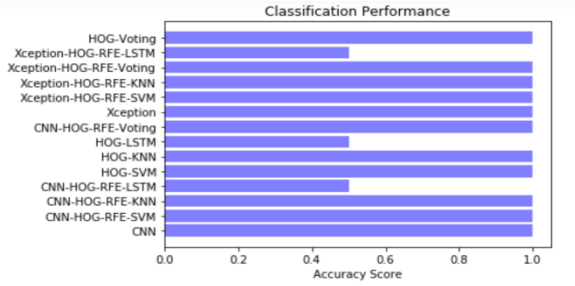

Fig 11 Accuracy graph

F1 Score: There is a machine learning rating tool called the F1 score that measures how accurate a model is. It adds up the accuracy and review scores of a model. The accuracy measurement figures out how often, across the whole collection, a model correctly predicted what would happen.

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

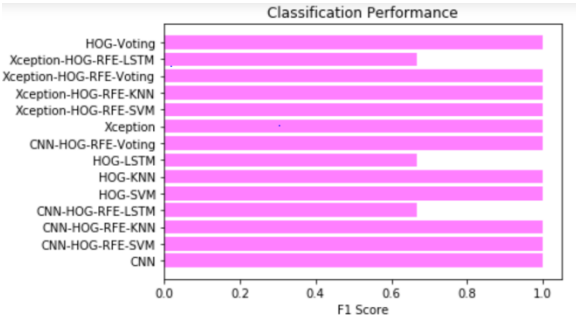

Fig 12 F1Score

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 13 Performance Evaluation table

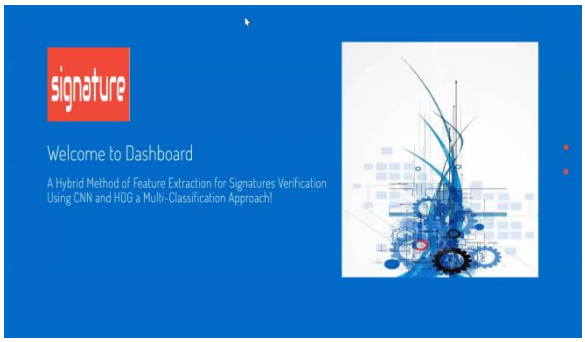

Fig 14 Home page

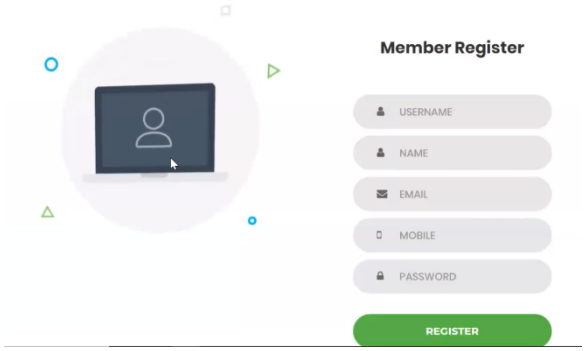

Fig 15 Registration page

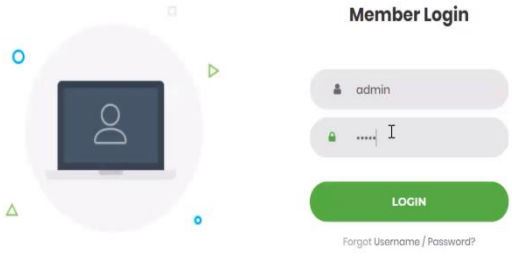

Fig 16 Login page

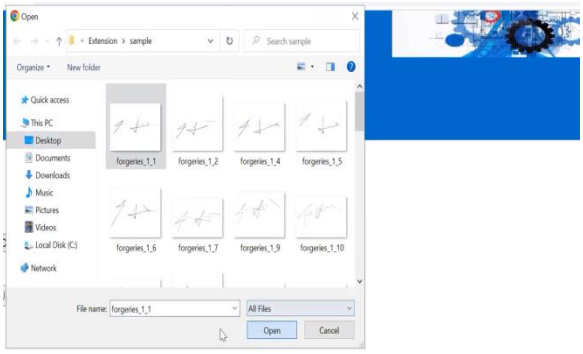

Fig 17 Input image folder

Fig 18 Upload input image

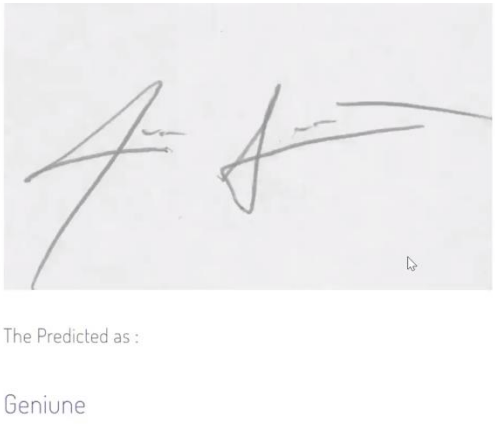

Fig 19 Predict result for given input

5. Conclusion

The For fast and accurate signature verification, the project proposes a combined technique using CNN and HOG.Optimization via decision trees ensures the mixed feature extraction approach works properly. Train the models using CNN, HOG, and Xception features to demonstrate the flexibility of the recommended strategy. SVM, KNN, and LSTM were selected as the best models because they accurately group signatures based on extracted features. Flask creates a simple interface for sharing and analyzing signature photos.Built-in user authentication makes the system more usable and secure. Xception, a decent dataset analysis tool, HOG-RFE feature extraction, and a Voting Classifier may score 100% [45]. Better speed and reliability make it an excellent CNN and HOG signature checker. A simple Flask interface improves system testing for data entry testers. Secure registration restricts system access to authorized users, making it safer.

6. FUTURE SCOPE

A very important part of verifying a signature is the process of feature extraction. You want to improve this process so that it better captures the unique features of signatures. This will make the checking method more accurate and trustworthy. The signature verification method should work better generally if the feature extraction step is made better. This includes making the system more accurate, lowering the number of false hits and blanks, and making it easier to tell if a signature is real or fake. [48] The signature proof method can be used for more things by making it work with mobile identification and e-signatures, for example. By being more flexible, the technology can be used in more safe entry points and meet a wider range of needs. Improving the user experience makes the system easy to use and available, which is important for getting more people to use it. Real-time reasoning is very important for things like security entry points and cash transactions. By making the model work better so that it gives quick and correct answers in real-time situations, it will be possible to use it in places where quick proof is important for safety and efficiency.

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/s00521-021-05854-6.

[3] A. Q. Ansari, M. Hanmandlu, J. Kour, and A. K. Singh, ‘‘Online signature verification using segment-level 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, ‘‘Off-line 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 off-line 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 back-propagation 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/understanding-lstmand-its-diagrams-37e2f46f1714