Enhancing Fingerprint Recognition Performance using KNN classifier and modified XG Boost Algorithm

Jainy Jacob M.
Department of Computer Applications, Mercy College Palakkad, India.

Abstract— **With the increasing use of biometric identification systems, there is a growing concern about the possibility of fake fingerprints being used to bypass security measures. Fingerprint recognition using KNN (K-Nearest Neighbors) algorithm incorporating modified optimization algorithm is an effective approach for biometric identification. KNN is a simple but powerful algorithm that can be trained on a dataset of fingerprint images and their corresponding labels to classify new images based on their nearest neighbors in the training set. The algorithm can be optimized by incorporating modified XGBoost, a gradient boosting algorithm, to improve its performance. The proposed approach of improving fingerprint recognition performance using a modified XGBoost algorithm and K-Nearest Neighbor classifier is a promising algorithm can effectively exploit the complementary strengths of both approaches and achieve superior performance in terms of accuracy and efficiency.**

Index Terms— **Gradient boosting, K-Nearest Neighbors, optimization algorithm, XGBoost, AdaBoost.**

I. INTRODUCTION

Fingerprint prediction from fake is the process of identifying whether a fingerprint is real or fake. With the increasing use of biometric identification systems, there is a growing concern about the possibility of fake fingerprints being used to bypass security measures. Fake fingerprints can be created using various methods, such as lifting a fingerprint from a surface or creating a synthetic fingerprint using materials like silicone or gelatin. The existing methods include analyzing the ridge and valley patterns of a fingerprint, detecting the presence of pores, and measuring the elasticity of the skin. Machine learning algorithms are often used to analyze these features and identify patterns that distinguish real fingerprints from fake ones. Thence, the proposed technique inclusive of K-Nearest Neighbor classifier incorporating modified XG Boost, performs fake prediction as an important area of research and development. It can help enhance the security of biometric identification systems. By detecting fake fingerprints, the risk of unauthorized access to sensitive information or facilities can be reduced, thereby improving overall security. As a significant of proposed research technique, it attains Enhanced security as fingerprints are unique to each individual, making it nearly impossible for someone else to access your device or account. In addition, it provides Accuracy with a low error rate as fast process and convenient, making it reliable and dependable on a multi-factor authentication, where a user's identity is verified using more than one method, making it more secure and non-intrusive.

II. LITERATURE SURVEY

This literature survey part provides a systematic and critical approach on acomprehensive overview of the research as described below.One of the algorithms used for fingerprint recognitions is SVM [3]. The Support Vector Machine (SVM) algorithm [4] employs a kernel function to transform the input space into a higher-dimensional feature space, eliminating the need to compute the inner product of two vectors. After the instances are mapped, the algorithm determines the optimal separating hyperplane with the maximum margin, thereby minimizing the upper bound of the expected risk instead of the empirical risk. To train SVMs, the Sequential Minimal Optimization (SMO) procedure is utilized. The C4.5 [5] algorithm is a machine learning method that builds decision trees to classify data. It generates the decision tree from a given set of examples in a top-down manner, extracting classification rules in the form of a tree structure. The algorithm selects an attribute at each node of the tree that maximizes the normalized information gain, which is determined by the difference in entropy, to split the data. This process is repeated recursively until the tree is fully grown. The decision tree is then pruned using the pessimistic pruning procedure to reduce overfitting. The k-Nearest Neighbours (kNN) [6] algorithm is classified as a lazy learning algorithm because it uses the entire training set as a reference set to classify new instances. The algorithm finds the group of k closest instances in the training set to the test instances and makes a decision based on the predominant class in this group. Thus, the distance metric and the number of neighbors to consider are crucial components of this method. To determine the optimal values for these parameters, a cross-validation procedure can be employed using the available training data. Jain et al. proposed a two-level multi-classifier approach [7] to simplify the classification problem by decomposing the original 5-class problem into 10 binary classification problems (all possible pairs of classes) using the One-vs-One (OVO) strategy [8,9]. They initially determine the two most probable classes of an input instance using a kNN algorithm with a large neighborhood (k = 10) before using a neural network to learn the classifiers in the second phase, where only the classifier considering both classes is utilized to output the final class. In their work, Nyongesa Nyongesa et al. [10] suggested utilizing three types of neural networks: multi-layer perceptron (MLP), radial basis function (RBF), and fuzzy neural network (FNN). Among these, MLP demonstrated the highest accuracy, leading us to choose it as the particular classifier for Nyongesa's feature vector. The MLP used in their study comprised of two hidden layers, with each layer having five nodes, and its weights were learned via the back-propagation learning technique. Shah and Sastry [11] presented three different classification approaches, among which we have opted for the most effective and general approach. In this method, the authors employ the support vectors of SVMs learned to distinguish between classes to identify the most significant fingerprints in the training set. These fingerprints are used as a reference set for the kNN algorithm (with k = 1), which extracts feature vectors from them for classifying new instances. Hong et al. proposed a classification approach that decomposes the problem using the One-vs-All strategy, where each binary problem distinguishes a class from the rest using SVMs. To break the tie when multiple classifiers give positive outputs, they dynamically order the SVMs using a Naïve Bayes classifier, whic h allows them to set an execution order and use the first positive output to label the instance. They use different features for each process, with FingerCode for SVMs and SPs and pseudo-ridge features for Naïve Bayes. Liu's method [12] uses ensembles of classifiers for fingerprint classification. Adaboost with logistic regression is used to learn multiple decision tree classifiers with fixed node numbers. Each part of the feature vector is learned with Adaboost, resulting in an ensemble of ensembles. The model is specific to fingerprint classification and uses OVA decomposition to address the multi-class problem. Leung and Leung [13] proposed a model for fingerprint retrieval, but it can also be used for classification. The missing values in the feature vector are corrected using mean imputation, and Fisher's Linear Discriminant Analysis is used for dimensionality reduction. The final classification is done with Quadratic Discriminant Analysis.

III. METHODS AND MATERIALS

The general steps for fingerprint recognition and classification in this proposed research technique is inclusive of pre-processing by standard technique following to Feature extraction by standard techniques. In prior to KNN classification, Feature scaling is done to get accurate result on classification. Scaling of the extracted features are ensured that they are in the same range. Commonly used feature scaling techniques are normalization and standardization. In Use of KNN algorithm, it classifies the fingerprint based on the selected features. In KNN, a new observation (fingerprint) is classified by comparing it with the K nearest observations in the training set. The class label of the new observation is assigned based on the majority vote of its K nearest neighbors. In sequence to KNN classifier,

Pre-Processing – The image is preprocessed to remove noise, enhance contrast, and normalize the image to a standard size and resolution after image acquisition. This step helps to improve the accuracy of fingerprint recognition. In sequence, feature Scaling is done via Standardscaler and transform. StandardScaler is a data preprocessing technique used in machine learning to standardize the features of a dataset. It is a commonly used method for feature scaling, which refers to the process of transforming the features of a dataset so that they have a similar scale or distribution. It is inclusive of the fit method for calculation of the mean and standard deviation of each feature and the transform method is then used to standardize the training and test data using the same parameters. This ensures that both the training and test data are scaled consistently [14].

K-Nearest Neighbors Classifier - In machine learning, the K-Nearest Neighbors (KNN) algorithm is a type of supervised learning algorithm that can be used for classification and regression tasks. In the case of fingerprint recognition and classification, the KNN algorithm can be used to classify new fingerprints based on their similarity to the preprocessed training data [15]. After preprocessing the fingerprint image and extracting relevant features, the next step is to create an instance of the KNeighbors Classifier class with the number of neighbors (n_neighbors) set to 4. The KNeighbors Classifier is a type of instance-based learning or lazy learning algorithm that stores all the training data during training and classifies new instances based on their similarity to the stored training data.

The algorithm works as follows

The preprocessed training data is used to create a search tree or a database structure to store the feature vectors and their corresponding class labels. During prediction, the feature vector of a new fingerprint is compared to the feature vectors of all the training fingerprints stored in the database. The KNN algorithm then selects the K nearest neighbors (i.e., K fingerprints with the smallest

Euclidean distance or any other distance metric) to the new fingerprint. The class label of the new fingerprint is assigned based on the majority vote of the K nearest neighbors. That is, the new fingerprint is assigned to the class label that appears the most among the K nearest neighbors.The choice of K (number of neighbors) can impact the performance of the KNN algorithm. If K is too small, the algorithm can be sensitive to noise and outliers. On the other hand, if K is too large, the algorithm can suffer from oversimplification and bias towards the majority class. The K-Nearest Neighbors (KNN) algorithm is a type of classification algorithm that can be used for fingerprint recognition.

The algorithm works as follows:

Let X be the preprocessed training data consisting of n feature vectors x1, x2, ..., xn, and their corresponding class labels y1, y2, yn.

Let x be the feature vector of a new fingerprint that needs to be classified.

During prediction, the Euclidean distance between x and each of the training feature vectors

x1, x2, xn is computed as:

Where xi represents the i-th training feature vector.

The K nearest neighbors of x is then selected based on the smallest distances.

Let N be the set of indices of the K nearest neighbors, i.e., N = {i1, i2, ..., iK}, where di1 ≤ di2 ≤ ... ≤ diK, for i = 1, 2, ..., n.

The class label of the new fingerprint is assigned based on the majority vote of the K nearest neighbors. That is, the new fingerprint is assigned to the class label that appears the most among the K nearest neighbors, i.e., y = argmax (yi, i in N) as K=4.

Thence, the KNeighbors Classifier is a type of instance-based learning algorithm that stores all of the training data and classifies new instances based on their similarity to the stored training data. It can be used for fingerprint recognition and classification by comparing the feature vector of a new fingerprint to the feature vectors of the preprocessed training fingerprints and assigning the class label based on the majority vote of the K nearest neighbors.

OPTIMIZATION – In sequence classifier, optimization is done to minimize the loss function, which is a measure of the difference between the predicted and true values of the target variable [16]. In optimization, the machine learning algorithm adjusts the model's

parameters or hyperparameters to improve its performance on the training data. There are many optimization algorithms, such as gradient descent, Adam, and Nesterov's accelerated gradient, that are used to find the optimal set of parameters or hyperparameters for a given model. Here in this proposed research, used a custom implementation of XGBoost with the Adam optimizer to train a gradient boosting model for regression as modified XGBoost.

The Adam optimizer is a type of optimization algorithm that is commonly used in gradient- based optimization methods such as gradient descent and gradient boosting. It is an adaptive learning rate

optimization algorithm that is designed to improve the convergence of the optimization process by adjusting the learning rate on a per-parameter basis [17]. In the context of gradient boosting, the Adam optimizer can be used to update the weights of the weak learners that are added to the model iteratively. During each iteration, the Adam
optimizer computes the gradient of the loss function with respect to the weights and updates the weights accordingly. The learning rate used by the optimizer is automatically adjusted based on the gradients observed during the optimization process.

Architecture of ADA boost in fingerprint Recognition

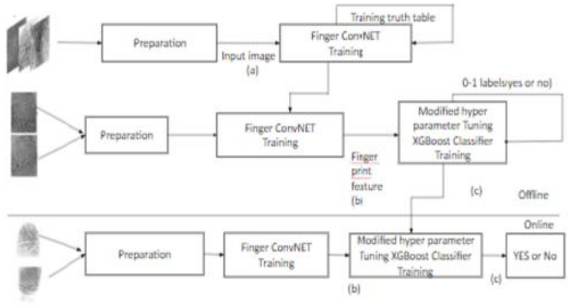


Figure 1

Algorithmic steps:
Initialize the model hyper parameters, including the learning rate, the number of trees, and the Adam optimizer parameters.
For each boosting round Compute the gradient and Hessian of the objective function with respect to the model predictions for the training data. Use the gradient and Hessian to update the model's weights and biases using the Adam optimizer update rule.
Evaluate the model's performance on the training and validation data using relevant metrics, such as the mean squared error.Optionally, prune the ensemble by removing weak or redundant trees based on their performance and the model's complexity.
Generate predictions for the test data using the final ensemble of trees. Evaluate the model's performance on the test data using relevant metrics, such as the Mean squared error.
The optimization process in a sequence classifier can be expressed using the following equation:
Initialize the model parameters, including the learning rate alpha, the decay rates beta1 and beta2, and the epsilon value for numerical stability.
Initialize the first and second moment estimates of the gradients to zero vectors:
m_0 = 0
v_0 = 0 For each boosting round t:
Compute the gradient and Hessian for the training data at the current round t: g_t = [Grad_i(X_t, y_t)] _{i=1} ^n is the gradient of the objective function w.r.t. the model
predictions for the training data at the current boosting round t, where X_t = [x_1, x_2, x_n] is the matrix of training inputs and y_t = [y_1, y_2, ..., y_n] is the vector of training targets.
H_t = [Hess_i(X_t, y_t)] _{i=1} ^n is the Hessian of the objective function w.r.t. the model predictions for the training data at the current boosting round t
Compute the first and second moment estimates of the gradients for the current round t:
m_t = beta1 * m_{t-1} + (1 - beta1) * g_t is the first moment estimate of the gradients,which is a running average of the previous gradients and the current gradient, where beta1is the exponential decay rate for the first moment estimates (default: 0.9).v_t = beta2 * v_{t-1} + (1 - beta2) * g_t^2 is the second moment estimate of the gradients,which is a running average of the previous squared gradients and the current squared gradient, where beta2 is the exponential decay rate for the second moment estimates (default: 0.999). Compute the bias-corrected first and second moment estimates of the gradients for the current round t: m_t_hat = m_t / (1 - beta1^t) is a bias-corrected first moment estimate, which accounts
for the initial bias of the estimates towards zero due to the initialization with zero vectors.v_t_hat = v_t / (1 - beta2^t) is a bias-corrected second moment estimate, which accounts for the initial bias of the estimates towards zero due to the initialization with zero vectors.
Update the model parameters using the Adam update rule:
delta_t = -alpha * m_t_hat / (sqrt(v_t_hat) + epsilon) is the update vector for the model parameters, where epsilon is a small positive value for numerical stability theta_t+1 = theta_t + delta_t is the updated vector of model parameters for the next round t+1.
Store the history of the first and second moment estimates of the gradients for each round t in a dictionary: results["Adam"] = custom.history Return the trained model with the updated parameters.

IV. RESULT

The sample input is taken is represented in figure 2 which is also represented as RGB format as execution style in figure 3 and made compared with figure 4 and sampled output is explored in figure 5

Figure 2

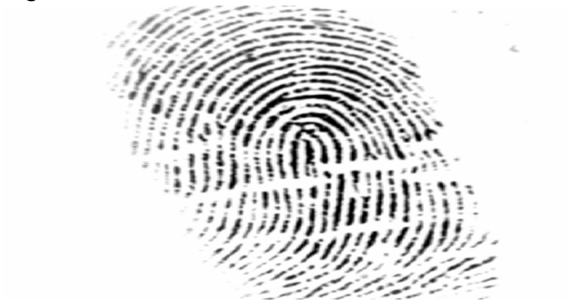


Figure 3

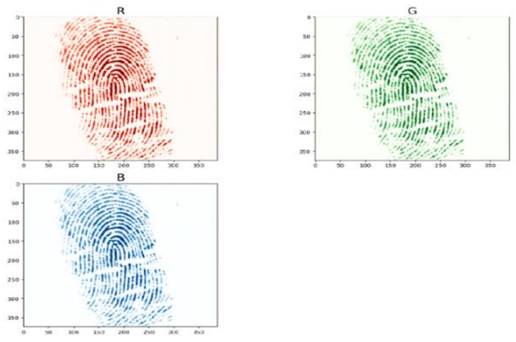


Figure 4

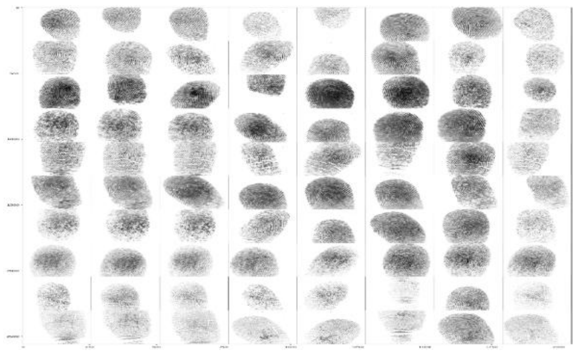
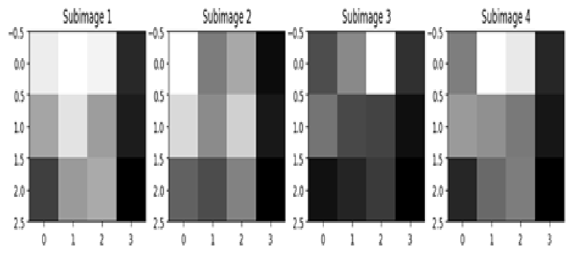
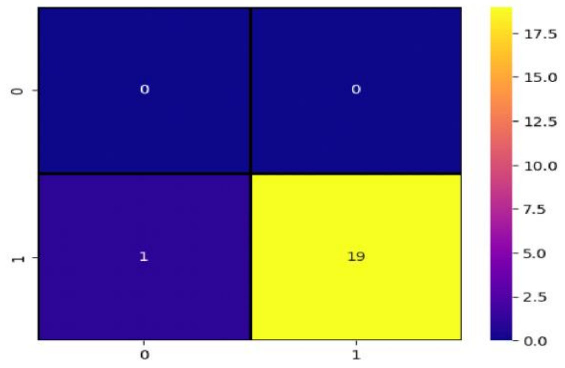


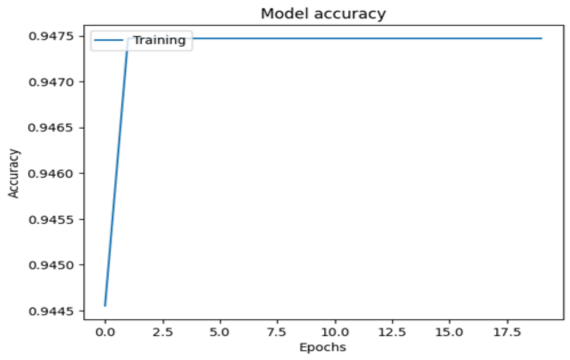
Figure 5

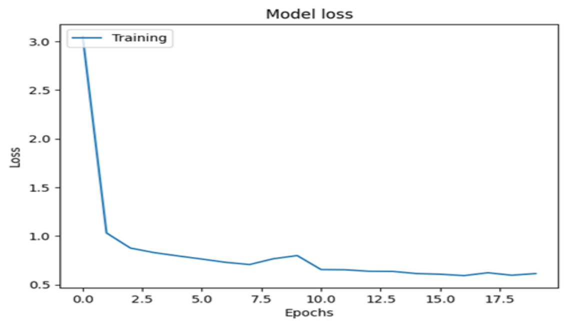


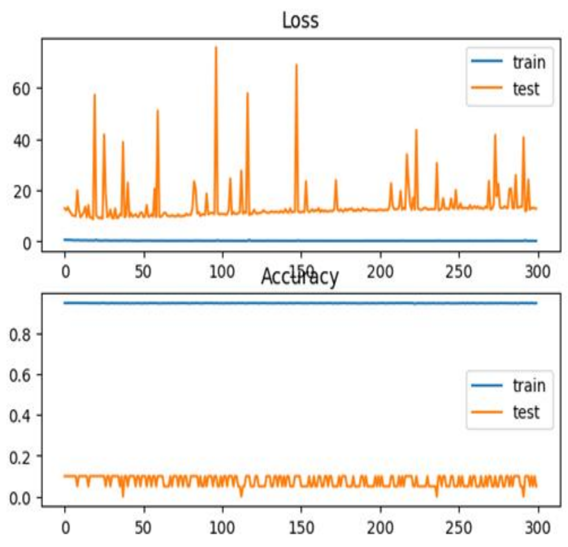
Evaluate the performance of the KNN model on the testing set via metrics such as accuracy,Precision, recall, and F1-score and confusion matrix [18] to assess its performance. The confusion matrix result obtained by KNN classifier is as

```
[[0 2 0 0 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0 1 0]
[0 1 0 1 0 0 0 0 0 0]
[0 0 0 0 1 1 0 0 0 0]
[0 0 1 0 1 0 0 0 0 0]
[0 0 0 0 1 1 0 0 0 0]
[0 0 0 0 0 0 1 0 0 1]
[0 0 0 0 0 0 0 1 0 1]
[0 0 1 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 2]]
```









IV. CONCLUSION

Classification on new fingerprint images and making predictions based on the features extracted incorporating modified XGBoost optimization into KNN for fingerprint recognition involves preprocessing the data, splitting it into training and testing sets, training the KNN model with modified XGBoost optimization, evaluating its performance, adjusting the hyper parameters, and using the model to make predictions on new data. Overall, KNN with modified XGBoost optimization is a robust and accurate approach for fingerprint recognition that can be applied in various real-world applications such as security systems, access control, and forensic investigations. The modified XGBoost algorithm uses the Adam optimizer for gradient boosting, which can improve the training speed and convergence of the algorithm compared to the traditional gradient boosting algorithm. The K-Nearest Neighbor classifier is used to refine the predictions of the XGBoost algorithm by incorporating the local similarity information of the fingerprint features. Experimental results on a publicly available fingerprint dataset show that the proposed approach achieves state-of-the-art performance in terms of recognition accuracy, with an average recognition rate of over 95% and a low false positive rate. The proposed approach also outperforms several other state-of-the-art fingerprint recognition methods, including deep learning-based methods, in terms of recognition accuracy and computational efficiency. the proposed approach of improving fingerprint recognition performance using a modified XGBoost algorithm and K-Nearest Neighbor classifier is a promising direction for future research in biometric recognition and security. The combination of these two algorithms can
effectively exploit the complementary strengths of both approaches and achieve superior performance in terms of accuracy and efficiency.

REFERENCES

[1] X. Fu, C. Liu, J. Bian, J. Feng, H. Wang and Z. Mao, "Extended clique models: A new matching strategy for fingerprint recognition", International Conference on Biometrics ICB 2013, pp. 1-6, 2013

[2] S. Gu, J. Feng, J. Lu and J. Zhou, "Latent fingerprint registration via matching densely sampled points", IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 1231-1244, 2021

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", Proceedings of the 32nd International Conference on Machine Learning ICML 2015, vol. 37, pp. 448-456, 2015

[4] Nivedita Soni and Ayasha Siddiqua, “Filtering Techniques used for Blurred Images in FingerprintRecognition”, International Journal of Scientific and Research Publications, vol 3, no 5, May 2013

[5] Ravi Kumar L, Sai Kumar S, Rajendra Prasad J, Subba Rao B.V and Ravi Prakash P, “Fingerprint Minutia Match Using Bifurcation Technique”, International Journal of Computer Science & Communication Networks, vol. 2, no. 4, pp. 478-486, Sep. 2012

[6] Jung, Ho Yub, Yong Seok Heo, and Soochahn Lee. "Fingerprint Liveness Detection by a Template-Probe Convolutional Neural Network." IEEEAccess 7(2019): 118986-118993

[7] Zhang, Yong, Dapeng Li, and Yujie Wang. "An indoor passive positioning method using CSI fingerprint based on Adaboost." IEEE Sensors Journal 19, no. 14 (2019): 5792-5800.

[8] Gu, Bin, Xin Quan, Yunhua Gu, Victor S. Sheng, and Guansheng Zheng. "Chunk incremental learning for cost-sensitive hinge loss support vector machine." Pattern Recognition 83 (2018):

196-208

[9] Wani, M. Arif, Farooq Ahmad Bhat, Saduf Afzal, and Asif Iqbal Khan. "Supervised Deep Learning in Fingerprint Recognition." In Advances in Deep Learning, pp. 111-132. Springer, Singapore, 2020.

[10] Uhl, Andreas. "State of the Art in Vascular Biometrics." In Handbook of Vascular Biometrics, pp. 3-61. Springer, Cham, 2020.

[11] Kim, Soowoong, Bogun Park, Bong Seop Song, and Seungjoon Yang. "Deepbelief network based statistical feature learning for fingerprint liveness detection." Pattern Recognition Letters 77(2016):58-65.

[12] [12] Belkhouja, Taha, Xiaojiang Du, Amr Mohamed, Abdulla K. Al-Ali, andMohsen Guizani. "Biometric-based authentication scheme for ImplantableMedical Devices during emergency situations." Future Generation Computer Systems 98 (2019): 109-119.

[13] Chan, Frodo Kin Sun, and Adams Wai Kin Kong. "A further study of lowresolution androgenic hair patterns as a soft biometric trait." Image and VisionComputing 69 (2018): 125-142.

[14] Malar, A. Christy Jeba, and Govardhan Kousalya. "Fingerprint-Based Support Vector Machine for Indoor Positioning System." In Smart Innovations inCommunication and Computational Sciences, pp. 289-298. Springer, Singapore, 2019

[15] Gómez, Jon Ander, Juan Arévalo, Roberto Paredes, and Jordi Nin. "End-to-endneural network architecture for fraud scoring in card payments." PatternRecognition Letters 105 (2018): 175-181.

[16] Belciug, Smaranda, and Florin Gorunescu. Intelligent Decision Support Systems-a Journey to Smarter Healthcare. Springer, 2020.

[17] Atri, Vinni. "Fingerprint Pre-processing and Features Extraction." Journal of Advancements in Library Sciences 5, no. 3 (2018): 40-45.

[18] Gupta, Phalguni, Kamlesh Tiwari, and Geetika Arora. "Fingerprint indexingschemes–A survey." Neurocomputing 335 (2019): 352-365

[19] Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Characterizingandevaluating adversarial examples for Offline Handwritten SignatureVerification." IEEE Transactions on Information Forensics and Security 14, no. 8 (2019): 2153-2166

[20] Ramya, R., and T. Sasikala. "An efficient Minkowski distance-based matchingwith Merkle hash tree authentication for biometric recognition in cloudcomputing." Soft Computing 23, no. 24 (2019): 13423-13431.