An Unsupervised Random Forest Approach to Spoofed Fingerprint Detection

Sean P. McKenna*, *Member, IEEE,* Laura R. Croft, Nathaniel J. Short, and Jonathan M. Levitt, *Member, IEEE*

***Abstract*—As the reliance on large fingerprint databases for various authentication and biometric applications grows, so does the need to maintain their credibility and security. Spoofed entries into such databases undermine their effectiveness and, consequently, a growing body of research is emerging that is directed at detecting spoofed entries. Various machine learning techniques such as neural networks and support vector machines have been used to identify spoofed fingerprints in a database. A critical drawback to these approaches is their need for training data. Arguably, in practice, appropriate training data may not exist, or if it does, the trained algorithm may not generalize to other fingerprint databases, rendering these approaches impractical. In this paper, we present an unsupervised approach that overcomes this shortcoming while still exhibiting comparable performance. Using an unsupervised random forest implementation, along with a simple input feature set, we demonstrate that spoofed fingerprint detection can be achieved without the need for training data. By using the dissimilarity matrix from the random forest as input to a hierarchical clustering scheme, we yield a classifier with equal error rates on the order of a few percent. As a result, this approach provides an attractive, viable, and much needed, alternative to traditional supervised methods of detection.**

***Index Terms*—Fingerprint recognition, spoofing, machine learning, random forest, classification algorithms.**

I. Introduction

FINGERPRINT recognition has long been used in law enforcement and forensics, and more recently, has gained prominence as a reliable, cost-effective means of biometric authentication. Government and commercial applications include immigration, border control, airport security, physical access control, ATM authentication, and mobile device security [1]. To quickly perform verification of a subject or to perform identification against a watch list, government agencies and other users of these systems maintain large databases of digital fingerprint records. Today, the FBI's Integrated Automated Fingerprint Identification System (IAFIS) contains records for over 64 million individuals [2].

Biometric systems are vulnerable to attacks at various stages in the biometric recognition process, including attacks on the database in which enrolled entries are stored. Large biometric databases pose challenges to testing and protecting the integrity of collected data. Fingerprint databases may be vulnerable to cyberattacks aimed at impersonating or concealing an individual's identity through the use of synthetically generated fingerprint images (spoofs). These images could allow the attacker to replace their own fingerprints in the database so that the attacker is not recognized during subsequent identification attempts. Additionally, an attacker can perform a "masquerade attack," in which they impersonate another individual by injecting a synthetic image that has been reconstructed from the desired individual's feature set [3]–[5].

S. P. McKenna is with Booz Allen Hamilton, 22 Batterymarch St., Boston, MA 02129 USA (e-mail: mckenna_sean@bah.com).

L. R. Croft, N. J. Short, and J. M. Levitt are with Booz Allen Hamilton, Suite 210, 1 Technology Dr., Westborough, MA 01581 USA.

Manuscript received July 8, 2014.

A. Motivation

A number of machine learning techniques (e.g., neural networks, support vector machines (SVMs), k-nearest neighbors) have been proposed to address the problem of spoofed fingerprint detection (e.g., [6]–[10]); however, each of these approaches requires training an algorithm, i.e., these approaches are all supervised techniques wherein a training dataset with known spoofed prints exists. These algorithms analyze the training data and produce an inferred function or model that can be used to analyze new samples. If such a training set does not exist or if extension of the results from a particular training set to a different database of prints fails, all of these techniques are rendered ineffective. Equally problematic is that trained algorithms are likely to have difficulty dealing with new, emerging methods of spoofing. An alternative approach that is not susceptible to these drawbacks is an unsupervised scheme that does not require a training set. Typically, unsupervised schemes fall into the class of clustering algorithms, as they seek to find the natural structure inherent in the input data. The k-means algorithm is a widely-used and computationally efficient unsupervised algorithm, but it often struggles with the disparate class size inherent in outlier detection problems such as spoofed fingerprint detection [11].

In this paper, we implement a traditionally supervised algorithm, the random forest (RF) [12], in an unsupervised manner as a basis for fingerprint spoof detection. RFs are a class of ensemble classifiers, growing many individual classification trees and aggregating the results. Each tree yields a classification, and the tree is said to "vote" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is grown as follows. If the number of cases in the dataset is N, N cases are sampled at random, with replacement. This sample is used to grow a tree. If there are M input variables (features), then at each node, a number $m \ll M$ is specified such that m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing (we found $m = 1$ to provide the most consistent results, which is not necessarily surprising given the small number of features we used). Each tree is grown to the largest extent possible and there is no pruning.

Some of the key benefits to RFs include: (1) Excellent accuracy among current algorithms, (2) efficiency on large databases (e.g., it can be parallelized), (3) the ability to provide estimates of which variables are important in the classification, (4) the calculation of proximities between pairs of cases that can be used in clustering and locating outliers, and (5) the ability to be extended to unlabeled data, leading to unsupervised clustering and outlier detection [12].

B. Outline

The remainder of this paper is organized as follows. Section II describes our methodology. We begin with the generation and acquisition of the fingerprint dataset we used, followed by a description of the feature set we chose for this analysis. Next, we explain how to implement an unsupervised RF, and how the resulting dissimilarity matrix can be used in a clustering scheme to separate the dataset into real and synthetic prints. We conclude this section with a test case that serves to verify our complete workflow. In Section III, we present our main findings. This includes an examination of how the number of trees and the number of spoofed prints affect performance. We then use the computed response operating characteristic and equal error rates to show that the proposed approach can achieve performance levels like those for supervised algorithms, which is an extremely encouraging result. Lastly, Section IV points out the significance of our results and discusses limitations and extensions of the proposed approach.

II. Approach

A. Fingerprint images

In this paper, our focus is to develop an scalable, automated technique to verify the integrity of a biometric database comprised of scanned fingerprint images. The Synthetic Fingerprint Generator (SFinGe, Biometric Systems Laboratory, University of Bologna, Cesana, Italy) [1], [13] was used to generate a set of 1000 synthetic fingerprint images, which served as spoof prints. The images were generated to mimic a sensor acquisition area of 13.0 mm $\times$ 17.1 mm with 500 pixels per inch (ppi) resolution (256×336 pixels image size). The SFinGe software tool first defines the external shape of the fingerprint area and the locations of distinguishing features such as loops and whorls. Based on the positions of those features, the ridge orientation pattern and the frequency density map are computed independently. From those criteria, the final pattern of fingerprint ridges and minutiae are generated using a contextual iterative filtering method. Finally, the software adds fingerprint-specific noise to produce a realistic-looking fingerprint image [13].

Four hundred real prints were obtained from a publicly available database [14] that were collected using an optical fingerprint scanner, the Cross Match Verifier 300 Classic, scanned at 500 ppi. To generate a dataset for classification, N_s synthetic prints were selected from the 1000 and randomly inserted into a group of $400 - N_s$ real prints. Thus, in all cases, the total fingerprint count was $N = 400$. Fig. 1 shows five prints from a sample dataset.

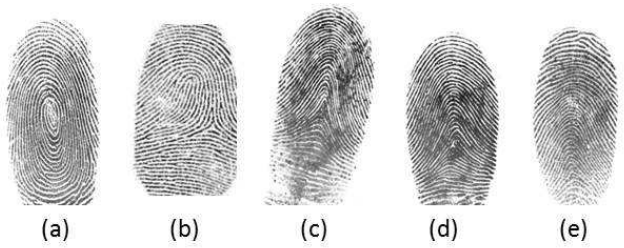

Fig. 1. Sample fingerprint images used in this analysis. Four are real prints and (b) is synthetic.

B. Feature selection

In order to use any sort of machine learning algorithm, supervised or otherwise, a set of features needs to be defined and collected. Such features can be discrete, continuous, or even categorical. In our analysis, we started with a simple set of features, and ultimately, found that such a set was sufficient to obtain favorable results. Specifically, we used seven first-order histogram features of each fingerprint image: energy, entropy, median, variance, skewness, kurtosis, and coefficient of variation. This group of features has been used by a number of others, e.g., [10], [15].

C. An unsupervised random forest

Traditionally, RFs are executed in a supervised mode–trained first, then run on the actual test data; however, there is a "trick" that permits RFs to execute as unsupervised learners [16]. To run a RF classifier in an unsupervised mode, one considers the original data as class 1 while creating a synthetic second class of the same size that is labeled as class 2. The synthetic second class is created by sampling at random from the univariate distributions of the original data. (For reference, this corresponds to `Addcl1` in the `R randomForest` package.) Thus, class 2 has the distribution of independent random variables, each one having the same univariate distribution as the corresponding variable in the original data. Class 2 destroys the dependency structure in the original data, but there are now two classes, and this artificial two-class problem can be run through a RF.

Aside from this one manipulation, there are typically only two parameters to consider when working with RFs: the number of trees (N_t) and a parameter usually referred to as `mtry`, which defines the number of feature variables sampled for splitting at each node. The default value for `mtry` is $\sqrt{M}$, which, in our case, is 2 or 3; however, we found a value of 1 to yield the most consistent performance for our RF. We consider the number of trees in Section III-B.

D. The dissimilarity matrix and clustering

Inherent in their generation, RFs compute a proximity measure between the observations. Concretely, the (i, j)-element of the proximity matrix produced by the RF is the fraction of trees in which elements i and j fall in the same terminal node. The interpretation is that "similar" observations should be in the same terminal nodes more often than dissimilar

ones. The proximity matrix can be used for for unsupervised learning with RFs. From the proximity matrix P, a distance, or dissimilarity matrix D is calculated as

$$D = \sqrt{1 - P}, \tag{1}$$

where, for the unsupervised case, these matrices are $2N \times 2N$. The dissimilarity matrix is then reduced in size (to $N \times N$) by eliminating all rows and columns corresponding to class 2. The resulting dissimilarity matrix can be taken as a distance measure, and clustering or multi-dimensional scaling using this distance matrix can be used to divide the original data points into groups for analysis and visual exploration. Specifically, we input D into an agglomerative hierarchical clustering scheme constrained to yield two clusters, which we consider as the separation of synthetic and real fingerprints.

We have implemented our approach in MATLAB using the open source `randomforest-matlab` port of the 'Random Forests' algorithm by Leo Breiman et al. from the `R` source. (We did explore the `R` package `randomForest`, but found it to be much slower and not practical for our purposes.) In addition, we make use of a handful of standard MATLAB toolbox functions to perform the clustering.

E. Test examples

As a way to verify, and gain confidence in, our approach, we considered a toy problem wherein an arbitrary dataset can be characterized by two features as specified below:

$$\begin{aligned} \text{feature 1} &= 1 + Cn_1 \\ \text{feature 2} &= 1 - Cn_2, \end{aligned}$$

where $C = 0.2$ and $n_i \sim \mathcal{N}(0,1)$, $i = 1, 2$. To mimic our fingerprint problem, we considered the situation where the total number of cases in the dataset was $N = 400$ and there were 40 "spoofs" or outliers ($N_s = 40$). The result was a 400×2 feature array where the first 40 rows consisted of cases having feature 1 and the remaining 360 rows had feature 2. The rows where then shuffled randomly. Such a two-cluster problem could easily be solved with a simple clustering algorithm (i.e., k-means), but it was used here as a check of our complete workflow. The output of our RF classifier, using 500 trees, was perfect binary classification. This can be visualized in Fig. 2, which shows the two distinct clusters after multidimensional scaling of the dissimilarity matrix.

Clearly, with such an idealized dataset, the clustering is trivial. To provide a sense of what the clustering looks like for our actual fingerprint dataset, which is encoded as a 400×7 feature array, we include Fig. 3. This figure shows the clustering after multidimensional scaling of the dissimilarity matrix produced from a 1000-tree RF. In this case, one can sketch a line that divides the data points into two clusters, which, while not as obvious as those in Fig. 2, are still apparent.

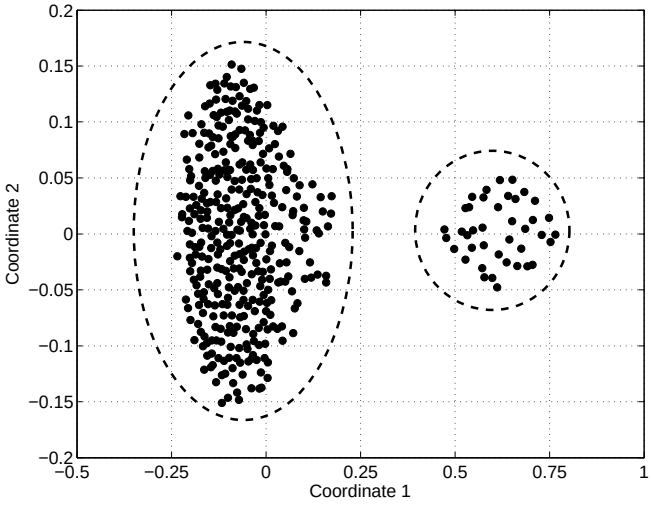

Fig. 2. Toy problem results showing two distinct clusters after multidimensional scaling of the dissimilarity matrix. There are 40 outliers, or "spoofs."

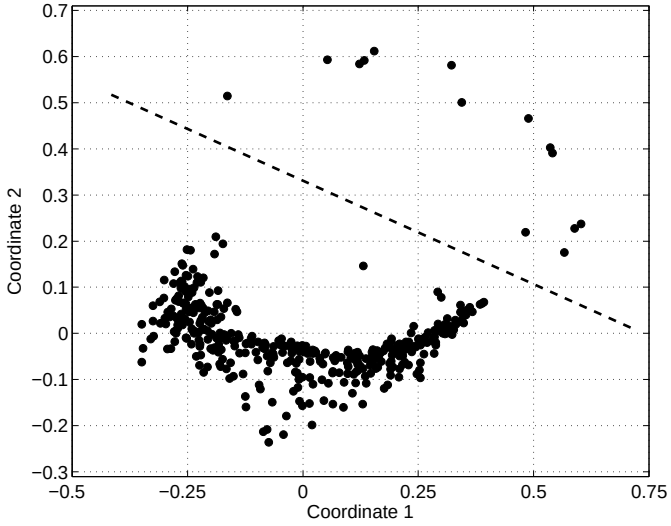

Fig. 3. Results from test data showing two clusters after multidimensional scaling of the dissimilarity matrix. There are 10 spoof prints and there are 14 points above the dashed line (drawn to delineate the clusters), indicating 4 false positives.

III. Results

A. Metrics of performance

To assess performance, we considered two metrics, the F-score (often used in the machine learning community) and the equal error rate (often used in the biometrics community). The F-score is the harmonic mean of the precision p and the recall r of the classifier:

$$\text{F-score} = \frac{2pr}{p + r} \tag{2}$$

and is a measure of a classifier's accuracy. The F-score attains its best value at 1 and worst score at 0. Both the precision and the recall are quantities derived from the elements of the confusion matrix, i.e.,

$$p = \frac{TP}{TP + FP} \tag{3}$$

and

$$r = \frac{TP}{TP + FN}, \tag{4}$$

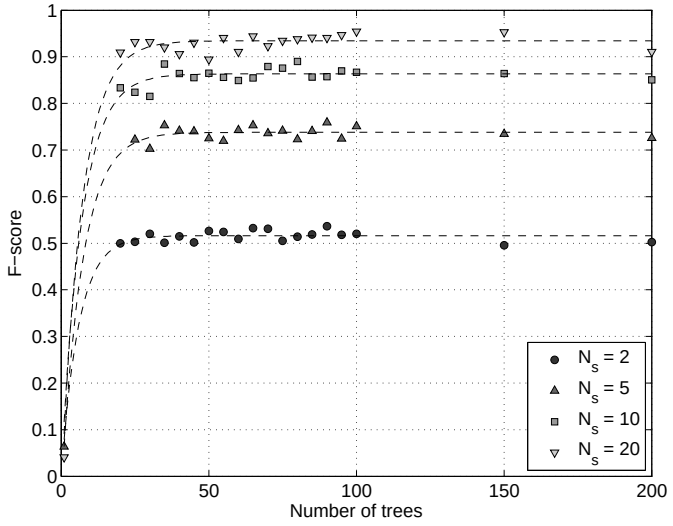

Fig. 4. Performance (F-score) as a function of number of trees and spoof count (N_s).

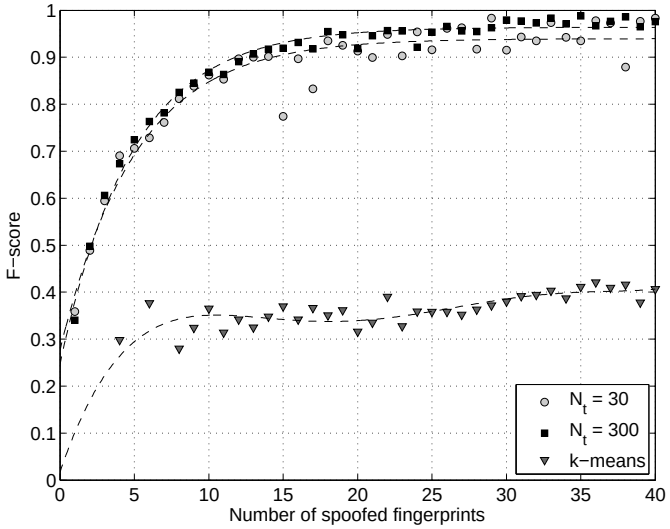

Fig. 5. Performance (F-score) as a function of number of spoofed fingerprints and tree count (N_t). Also included is a k-means result.

where TP is true positive ("hit"), FP is false positive ("false alarm"), and FN is false negative ("miss"). High precision indicates that an algorithm returned substantially more relevant results than irrelevant, while high recall indicates that an algorithm returned most of the relevant results [17].

The equal error rate (EER) is the value when the false acceptance rate (FAR) and the false rejection rate (FRR) are equal. The FAR is the probability of accepting a spoofed fingerprint as a real one, and the FRR is the probability of rejecting a real fingerprint as a spoofed one, i.e.,

$$\text{FAR} = \frac{FP}{N_s} \tag{5}$$

and

$$\text{FRR} = \frac{FN}{N - N_s}. \tag{6}$$

In evaluating a classifier, the smaller the EER, the better.

B. Influence of the number of trees

In general, when using a RF to arrive at a dissimilarity measure, the more trees that are grown, the better the dissimilarity measure will be able to separate observations. In our early analysis, we typically used thousands of trees, but it became evident that such a large number of trees was not necessarily required—at least in terms of the F-score metric. Fig. 4 illustrates this finding by showing the F-score as a function of the number of trees for four spoof counts. Clearly, beyond approximately 30 trees, the F-score is essentially unchanged. This does not necessarily imply that a greater number trees will not improve performance (c.f., Fig. 6, for instance); however, if computational considerations are a concern, then selecting the number of trees to use becomes important, and results like those in Fig. 4 can help guide that decision.

C. Influence of the number of spoofed fingerprints

As our goal of spoof detection ultimately becomes a clustering problem, the ratio of the number of spoofed fingerprints to total number of prints is a critical factor. This is borne out by the results shown in Fig. 5, which presents the F-score as a function of the number of spoofed fingerprints for two tree counts. Before discussing the role of the number of spoofed prints, it is worth noting that, consistent with Fig. 4, increasing the number of trees by an order of magnitude has very little effect of the F-score. On the other hand, the number of spoofed prints has a significant effect on the F-score. To achieve an F-score greater than 0.9, one needs roughly 15 spoofs in the dataset, which corresponds to a spoof density of 3.75%. Below this value, the F-score drops off rapidly. As a point of comparison, Fig. 5 includes a result from running a k-means clustering (with cosine distance metric) on the data. As might be expected, the k-means F-score is much lower than that of the RF scheme, owing to the class imbalance.

D. Receiver operating characteristic

Since there does not exist a typical threshold-type parameter in the proposed RF clustering approach that would lend itself to generating a receiver operating characteristic (ROC) curve, we used the spoof count in lieu of a threshold parameter. In this way, we produced the ROC curves shown in Fig. 6, which show the true acceptance rate ($1-$FRR) as a function of the false acceptance rate (FAR) for three tree counts. Considering the results in this manner, we observe that there is a benefit to using a larger number of trees, although diminishing returns seem to hold as the number of trees gets large (thousands). Using the ROC curves, we estimated the following EER values: 2.3% for 3000 trees, 2.7% for 300 trees, and 6.0% for 30 trees. These values compare well with other *supervised* approaches (e.g., [9], [15]).

IV. Conclusion

In this paper, we have demonstrated an automated technique to verify the integrity of a biometric database. We have shown

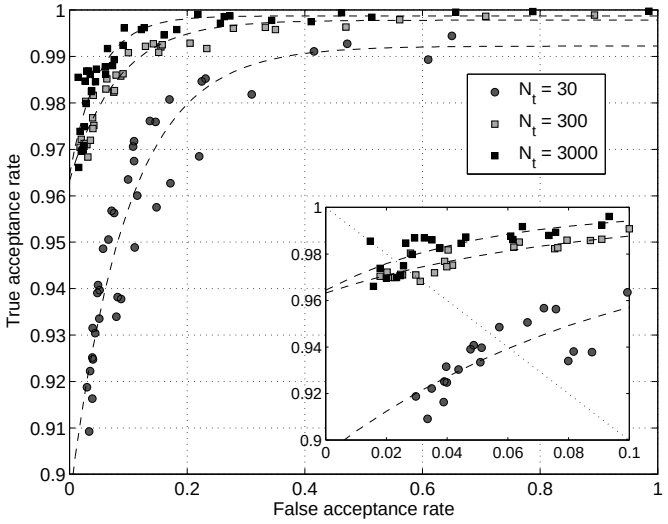

Fig. 6. The true acceptance rate (1−FRR) as a function of the false acceptance rate (FAR) for three tree counts. The inset shows the same relationship, rescaled such that the EER line (dotted) can be seen.

that a significant drawback of current supervised learning approaches (i.e., the need for training data) to spoofed fingerprint detection can be avoided through the use of an unsupervised random forest approach. With a very limited input feature set and resulting EER values near 2%, this scheme shows significant promise and should open the door for other unsupervised approaches such as self-organizing maps (e.g., [18]) and unsupervised SVMs (e.g., [19]). Furthermore, while this study has focused on synthetic spoofs, it has the potential to be extended to the detection of spoofs created using gelatins, glues, and other materials. It also has the benefit of being relatively straightforward to parallelize, thereby enabling it to operate on large databases.

As noted, the features used in this approach were rather primitive and their success was likely due to the high-quality set of prints used (synthetic and scanned). For more challenging situations such as ink fingerprints and other types of spoofed prints, it is likely that additional features (e.g., [8], [9], [15]) would be needed to enable the dataset to be accurately separated into spoofed and real prints. Another point worth mentioning is that some fingerprint databases do not store the actual prints themselves, but instead store only the minutiae template. It would be worthwhile to test the proposed algorithm on such a database.

REFERENCES

[1] D. Maltoni and R. Cappelli, "Fingerprint Recognition," in *Handbook of Biometrics*, 2008, pp. 23–42.

[2] K. R. Moses, P. Higgins, M. McCabe, S. Prabhakar, and S. Swann, "Automated fingerprint identification systems (AFIS)," in *National Institute of Justice/NCJRS*, 2010, pp. 6:1–33.

[3] N. Ratha, J. Connell, and R. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, 2001.

[4] R. Cappelli, D. Maio, A. Lumini, and D. Maltoni, "Fingerprint image reconstruction from standard templates," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 9, pp. 1489–1503, Sept 2007.

[5] A. Ross, J. Shah, and A. Jain, "From template to image: Reconstructing fingerprints from minutiae points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 4, pp. 544–560, April 2007.

[6] R. Derakhshani, "Spoof-Proofing Fingerprint Systems Using Evolutionary Time-Delay Neural Networks," in *Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety*, no. April, 2005.

[7] S. Nikam and S. Agarwal, "Fingerprint anti-spoofing using ridgelet transform," in *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, Sept 2008, pp. 1–6.

[8] T. Barsky, "Fingerprints image spoof detection and classification," Master's thesis, Tel-Aviv University, 2009.

[9] B. Tan and S. Schuckers, "Spoofing protection for fingerprint scanner by fusing ridge signal and valley noise," *Pattern Recognition*, vol. 43, no. 8, pp. 2845–2857, 2010.

[10] L. Pereira, H. Pinheiro, J. Silva, A. Silva, T. Pina, G. D. C. Cavalcanti, T. I. Ren, and J. de Oliveira, "A fingerprint spoof detection based on mlp and svm," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, June 2012, pp. 1–7.

[11] J. Liang, L. Bai, C. Dang, and F. Cao, "The k-means-type algorithms versus imbalanced data distributions," *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 4, pp. 728–745, Aug 2012.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[13] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, "Synthetic fingerprint generation," *Handbook of Fingerprint Recognition*, pp. 271–302, 2009.

[14] Neurotechnology, "Sample fingerprint databases," 2013, www.neurotechnology.com/download.html#databases.

[15] H. Choi, R. Kang, K. Choi, A. T. B. Jin, and J. Kin, "Fake-fingerprint detection using multiple static features," *Optical Engineering*, vol. 48, no. 4, pp. 047 202:1–13, Apr. 2009.

[16] T. Shi and S. Horvath, "Unsupervised learning with random forest predictors," *Journal of Computational and Graphical Statistics*, vol. 15, no. 1, pp. 118–138, 2006.

[17] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240.

[18] T. Kohonen and T. Honkela, "Kohonen network," *Scholarpedia*, vol. 2, no. 1, p. 1568, 2007, revision #122029.

[19] S. Winters-Hilt and S. Merat, "SVM clustering," in *Proceedings of the Fourth Annual MCBIOS Conference – Computational Frontiers in Biomedicine*, D. Wilkins, Y. Gusev, R. Loganantharaj, S. Bridges, S. Winters-Hilt, and J. D. Wren, Eds., vol. 8(Suppl 7), Nov. 2007, p. S18.