




# Fingerprint Large Classification Using Sequential Learning on Parallel Environment

Nicolás A. Reyes-Reyes<sup>1</sup><sup>a</sup>, Marcela C. González-Araya<sup>2</sup><sup>b</sup> and Wladimir E. Soto-Silva<sup>3</sup><sup>c</sup>

<sup>1</sup>*Programa de Doctorado en Sistemas de Ingeniería, Facultad de Ingeniería, Universidad de Talca, Campus Curicó, Camino a Los Niches km 1, Curicó, Chile*

<sup>2</sup>*Departamento de Ingeniería Industrial, Facultad de Ingeniería, Universidad de Talca, Campus Curicó, Camino a Los Niches km 1, Curicó, Chile*

<sup>3</sup>*Departamento de Computación e Industrias, Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Avenida San Miguel 3605, Talca, Chile*

**Keywords:** Fingerprint, Large Classification, Sequential Learning, Extreme Learning Machine, Graphics Processing Unit.

**Abstract:** Fingerprint classification allows a biometric identification system to reduce search space in databases and therefore response times. In the literature, fingerprint classification has been addressed through different approaches where deep learning techniques such as convolutional neural networks have been gaining attention. However, the proposed approaches use extremely small data sets for large-scale real-world scenarios that could worsen accuracy rates due to interclass and intraclass variations in fingerprints. For this reason, we proposed a fingerprint classification approach that allows us to address this problem by considering millions of samples. For this purpose, a classifier based on neural networks trained using online sequential extreme learning machines was developed. Likewise, to accelerate the training of the classifier, the matrix operations inside it was run in a graphic processing unit. In order to evaluate our proposal, the approach was tested on three datasets with more than two million synthetic fingerprint image descriptors. The results are similar in terms of accuracy and computational time to recent approaches but using more than 2.5 million samples.


## 1 INTRODUCTION


One of the most widely used biometric techniques for identifying people today corresponds to the fingerprint. This is because it contains the necessary information about unique characteristics of a person (Zia et al., 2019). The fingerprint has applications in many areas, and is mainly used in security and control systems such as entry to mass events, police control, national registration of people, assistance in companies, etc (Galar et al., 2015a; Zabala-Blanco et al., 2020b).


In a fingerprint recognition system, to determine a person's identity, their fingerprint is searched in a database until a match is found. However, this is quite computationally expensive if the database is large. For this reason, a fingerprint classification is commonly performed, reducing the domain or search space in the database. Fingerprint images have structural characteristics based on the pattern

of the ridges, and according to their morphological structure they are usually classified into five classes (known as Henry's classification), which are the following: Arch, Left Loop, Right Loop, Tented Arch and Whorl. Furthermore, these classes present a high level of imbalance between them, like other human biological characteristics (Peralta et al., 2018; Saeed et al., 2018; Zabala-Blanco et al., 2020b).

Several approaches have been presented in the literature to solve this challenging fingerprint classification problem. For instance, Rajanna et al. (2010); Cao et al. (2013) and Luo et al. (2014) proposed classifiers based on k nearest neighbors (kNN) and support vector machines (SVM), and assembled as hierarchical classifiers. Furthermore, to improve the performance of the classifiers in terms of accuracy, the authors used feature extractors such as orientation maps, curvelet transform along with gray level co-occurrence matrices, among others. Rule-based classifiers have also been applied by Liu (2010); Guo et al. (2014); Galar et al. (2014) and Dorasamy et al. (2015) as decision trees (DT), and these in combination with adap-

<sup>a</sup> <https://orcid.org/0009-0002-1100-6187>

<sup>b</sup> <https://orcid.org/0000-0002-4969-2939>

<sup>c</sup> <https://orcid.org/0000-0002-2203-9366>



tive boosting (AdaBoost). Likewise, to achieve better classification results, feature extractors based on singular points, orientation maps and directional patterns were used. On the other hand, one of the most used techniques to classify fingerprints is SVM. Thus, studies presented by Cao et al. (2013); Saini et al. (2013); Galar et al. (2015b); Gupta and Gupta (2015); Huang et al. (2015) and Alias and Radzi (2016), presented SVMs as single classifiers and in combination with other techniques such as probabilistic neural networks (PNN), and also as part of a hierarchical classifier. In this case, the authors improved the performance of the classifiers by using feature descriptors based on orientation fields, designs based on Wavelet transforms, singular points, minutiae extraction, and Fourier spectrum features. Also, classifiers have been proposed based on adaptive genetic neural networks (AGNN) by Borra et al. (2018), on naive-bayes by Vitello et al. (2014), and on conditional probability by Jung and Lee (2015).

In recent years, deep learning (DL) techniques have focused the attention of researchers for fingerprint classification. Hence, multiples convolutional neural networks (CNN) have been proposed by Wang et al. (2016); Michelsanti et al. (2017); Ge et al. (2017); Shrein (2017); Peralta et al. (2018); El Hamdi et al. (2018), and Zia et al. (2019), considering different amounts of convolutional layers, or auto-encoder (AE) layers, and even pretrained networks like VGG-F and VGG-S. In this case, CNNs calculate feature descriptors through internal mechanisms, and other techniques were not required by the authors. However, CNNs require large datasets to achieve high accuracy and high computational effort to train due to their complexity. For this reason, neural networks trained using extreme learning machines (ELM) are an alternative to CNNs since they can achieve similar accuracy rates and their training is extremely fast (Huang et al., 2006). Thus, ELM neural networks have been proposed as a classifier in human recognition from images by several studies (An et al., 2015; Li et al., 2015; Zhang et al., 2018; Deng et al., 2019; Lu et al., 2019). Specifically, it has been used for fingerprint classification by Saeed et al. (2018) in its basic form with radial activation function, considering the extraction of features in fingerprint images using orientation fields with histograms of oriented gradient (HOG). Also, Zabala-Blanco et al. (2020b) proposed classifiers based on these neural networks in their basic and weighted form to treat unbalanced classes. These authors used different feature extractors such as Hong08, Liu10, and Capelli02, which are the most common extractors currently for fingerprint classification.

Although the approaches presented in the literature have achieved high accuracy rates (even above 0.95) and training times less than 5,000 seconds (mostly), the datasets used in them are extremely small. Furthermore, almost all of them resort to publicly accessible datasets such as NIST-DB4 OR FVC2000, 2002, 2004 that do not exceed 4,000 fingerprint samples. Sometimes the SFinGe software was used to generate synthetic fingerprints, but these do not exceed 30,000 samples either. These conditions are limiting for the classification of fingerprints in biometric systems intended for large populations of people because the fingerprints that best represent each class must be chosen before performing the classification. Therefore, this choice is subject to human error even though it is carried out by experts due to inter-class similarities and intra-class differences in fingerprints (Zia et al., 2019; Zabala-Blanco et al., 2020b). For this reason, in this study we propose an approach for large fingerprint classification in reasonable computational times for practical scenarios. For this purpose, a classifier based on neural networks trained using the online sequential extreme learning machine (OS-ELM) is developed. Furthermore, training of the classifier is accelerated by running its matrix operations to a graphics processing unit (GPU). Additionally, specialized feature extractors are used to obtain input data from synthetic fingerprint images. To validate the proposal, this approach is applied on three data sets with millions of fingerprint descriptors.

The rest of this document is organized as follows. Section 2 describes the proposed approach for large fingerprint classification using neural networks and sequential learning. Section 3 reports and discusses the results of computational experiments carried out on three fingerprint descriptor datasets. Finally, the 4 section presents the main conclusions of this study and future work.

## 2 PROPOSED APPROACH

The proposed approach for large fingerprint classification is summarized in Figure 1.

Each stage of the approach is briefly described in the following subsections.

### 2.1 Large Datasets Generation

Three feature descriptor datasets are generated from synthetic fingerprint images. To generate fingerprint images with characteristics and distribution that simulate real fingerprints, the SFinGe v4.0 software is used. The Capelli02, Hong08 and Liu10 extractors



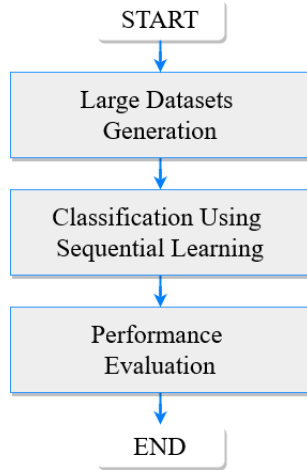


Figure 1: Flow of the proposed approach for fingerprint classification.

Table 1: Distribution of fingerprint classes in each dataset.

Fingerprint class	No. of samples
Arch	660,573
Left Loop	5,947,535
Right Loop	5,599,876
Tented Arch	515,757
Whorl	4,973,608

are used to calculate the feature descriptors on the generated syntactic fingerprint images. These extractors are used since the descriptors they generate have a high quality and capacity to represent the most important visual characteristics of a fingerprint image (Zabala-Blanco et al., 2020b). Each dataset has a total of 17,697,349 fingerprint descriptors, and each descriptor is labeled. The classes considered are: Arch, Left Loop, Right Loop, Tented Arch and Whorl, as shown in the following Figure 2.

The distribution of these five classes of fingerprints is not uniform in the three datasets, representing the following percentages of the total samples: Arch (3.73 %), Left Loop (33.61 %), Right Loop (31.64 %), Tented Arch (2.91 %) and Whorl (28.10 %) (as shown in Table 1).

These datasets are used considering balanced classes to achieve better classification performance. This balancing is carried out through random subsampling.

## 2.2 Classification Using Sequential Learning

The extreme learning machine (ELM) neural network is a feed-forward network with a single hidden layer (SLFN), with its respective input and output layer.

This neural network is classified as a network with random weights. The training of this network consists of randomly assigning the weights and biases of the hidden layer, and analytically computing the weights between the hidden layer and the output layer (Huang et al., 2006, 2015; Cao et al., 2018).

In Huang et al. (2006), this ELM neural network is formally defined as follows:

Given  $N$  different arbitrary samples  $(\mathbf{x}_i, \mathbf{t}_i)$ , where  $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$  and  $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbb{R}^m$ , the standard single hidden layer feed-forward neural network with  $L$  neurons and activation function  $g(x)$  is mathematically modeled as:

$$\sum_{i=1}^L \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^L \beta_i g(\mathbf{w}_i * \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \dots, N \quad (1)$$

where  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$  is the vector of weights connecting the  $i$ -th hidden neuron and the input neurons,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the vector of weights connecting the  $i$ -th hidden neuron and the output neurons, and  $b_i$  is the bias of the  $i$ -th hidden neuron.  $\mathbf{w}_i * \mathbf{x}_j$  denotes the inner product of  $\mathbf{w}_i$  and  $\mathbf{x}_j$ . The output neurons are chosen linear.

The standard single-layer feed-forward neural network hidden with  $L$  neurons and with activation function  $g(x)$  can approximate these  $N$  samples with zero mean error  $\sum_{j=1}^N \|\mathbf{o}_j - \mathbf{t}_j\| = 0$ , that is, they exist  $\beta_i, \mathbf{w}_i$  and  $b_i$  such that:

$$\sum_{i=1}^L \beta_i g(\mathbf{w}_i * \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \dots, N \quad (2)$$

The  $N$  equations above in (2) can be written in compact form as:

$$\mathbf{H}\beta = \mathbf{T} \quad (3)$$

where the matrix  $\mathbf{H}$  is:

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_L, b_1, \dots, b_L, \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 * \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_L * \mathbf{x}_1 + b_L) \\ \vdots & \dots & \vdots \\ g(\mathbf{w}_1 * \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_L * \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L}, \quad (4)$$

the vector of weights between hidden layer and output layer:

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad (5)$$

and the vector of labels or targets of the  $N$  samples is:

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \quad (6)$$





Figure 2: Fingerprint classes.

The matrix  $\mathbf{H}$ , is the output matrix of the hidden layer of the neural network. The  $i$ -th column of  $\mathbf{H}$  is the output of the  $i$ -th hidden neuron with respect to the inputs  $x_1, x_2, \dots, x_N$ .

Considering that both  $\mathbf{H}$  as  $\mathbf{T}$  are known, the weight vector between the hidden layer and the output layer  $\beta$ , is determined by:

$$\beta = \mathbf{H}^\dagger \mathbf{T} \quad (7)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose inverse or also known as the pseudoinverse of the matrix  $\mathbf{H}$  (Greville, 1959). This generalized inverse can be obtained by means of the following expression.

$$\mathbf{H}^\dagger = (\mathbf{H}^T * \mathbf{H})^{-1} * \mathbf{H}^T \quad (8)$$

In addition to equation (8) (known as orthogonal projection), there are different algorithms that allow calculating the generalized inverse of a matrix, among these are algorithms based on singular value decomposition (SVD), extended orthogonal projection (with regularization parameter) (EOP), Cholesky factorization, among others (Golub and Kahan, 1965; Courrieu, 2008; Lu et al., 2015). Specifically, the EOP algorithm performs a fast and effective calculation of the generalized inverse, achieving a high level of precision (Lu et al., 2015). For this reason, this algorithm is useful in training an ELM neural network. The EOP is defined as the equation (9) below:

$$\mathbf{H}^\dagger = \begin{cases} (\mathbf{H}^T * \mathbf{H} + \frac{\mathbf{I}}{C})^{-1} * \mathbf{H}^T, & \text{if } N \geq L \\ \mathbf{H}^T * (\mathbf{H} * \mathbf{H}^T + \frac{\mathbf{I}}{C})^{-1}, & \text{otherwise} \end{cases} \quad (9)$$

where  $\mathbf{I}$  and  $C$  in (9), correspond to an identity matrix and a regularization parameter respectively.

Several extensions of the basic ELM have been developed, with features to solve different types of problems and/or scenarios (Huang et al., 2015). One of these extensions is the online sequential extreme learning machine, which is described below.

### 2.2.1 Sequential Learning

The online sequential extreme learning machine (OS-ELM) is a feed-forward neural network training algorithm, based on the standard ELM and the recursive

least squares (RLS) algorithm. Where the main difference with the standard, is that the OS-ELM does not need to process all the data at once, since it can perform the training by processing smaller batches of data and update the output weights of the hidden layer in the neural network iteratively. In OS-ELM, the data batches to be processed can maintain a fixed size (equal number of samples per batch) during training, and can even vary in size, making the training process even more flexible (Liang et al., 2006).

The OS-ELM achieves excellent generalizability in a feed-forward neural network, and in extremely short times, thus preserving the fundamental properties of the standard ELM (Liang et al., 2006; Huang et al., 2015).

Algorithm 1 describes the procedure associated with the OS-ELM:

To strengthen the calculation of the inverse matrix in the equation (10) (in Algorithm 1), the calculation of a usual inverse can be replaced by a generalized or pseudoinverse (equation 9). This allows to avoid the distortion produced in the inverse of a matrix, due to the proximity of the original matrix to a singularity condition (Liang et al., 2006; Huang et al., 2015).

## 2.3 Performance Evaluation

To evaluate performance, a simple cross-validation scheme is applied. In this simple cross-validation scheme, each original dataset is divided into two subsets, one to perform the training and the other to test the trained classifier and verify its performance, with new data. In some cases, a third subset is considered to find good values for the hyperparameters of a classifier (Raschka, 2018). Therefore, each dataset is finally divided in a 80:20 ratio, obtaining a subset for training-validation (with 80% of the total samples) and a subset for testing (with 20% of the total samples), as performed by Peralta et al. (2018) and Zabala-Blanco et al. (2020b).

Classification performance is measured through accuracy (ACC) for all the experiments associated with the three datasets. These performance measures



**Require:** Initial training data block  $\mathcal{E}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$  of a given training set  $\mathcal{E} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \dots\}$ , an activation function  $g(x)$  and a number of neurons  $L$ , where  $N_0 \geq L$ .

### 1. Initialization Phase:

**Step 1:** Randomly assign input weights and biases  $\mathbf{w}_i, b_i, i = 1, \dots, L$ .

**Step 2:** Calculate initial output matrix of hidden layer  $\mathbf{H}_0$ , where

$$\mathbf{H}_0 = \begin{bmatrix} g(\mathbf{w}_1 * \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_L * \mathbf{x}_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 * \mathbf{x}_{N_0} + b_1) & \cdots & g(\mathbf{w}_L * \mathbf{x}_{N_0} + b_L) \end{bmatrix}_{N_0 \times L}$$

**Step 3:** Estimate the initial output weights  $\beta^0 = \mathbf{P}_0 * \mathbf{H}_0^T * \mathbf{T}_0$  where  $\mathbf{P}_0 = (\mathbf{H}_0^T * \mathbf{H}_0)^{-1}$  and

$$\mathbf{T}_0 = [\mathbf{t}_1, \dots, \mathbf{t}_{N_0}]^T.$$

**Step 4:** Set  $k = 0$ .

### 2. Sequential Learning Phase:

Present the  $(k + 1)$ -th batch of data

$$\mathcal{E}_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^k N_j)+1}^{\sum_{j=0}^{k+1} N_j}$$

where  $N_{k+1}$  denotes the number of samples in the batch  $k + 1$ .

**Step 1:** Calculate partial output matrix of hidden layer  $\mathbf{H}_{k+1}$  for batch data  $\mathcal{E}_{k+1}$ .

$$\mathbf{H}_{k+1} = \begin{bmatrix} g(\mathbf{w}_1 * \mathbf{x}_{(\sum_{j=0}^k N_j)+1} + b_1) & \cdots & g(\mathbf{w}_L * \mathbf{x}_{(\sum_{j=0}^k N_j)+1} + b_L) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 * \mathbf{x}_{\sum_{j=0}^{k+1} N_j} + b_1) & \cdots & g(\mathbf{w}_L * \mathbf{x}_{\sum_{j=0}^{k+1} N_j} + b_L) \end{bmatrix}_{N_{k+1} \times L}$$

**Step 2:** Set  $\mathbf{T}_{k+1} = [\mathbf{t}_{(\sum_{j=0}^k N_j)+1}, \dots, \mathbf{t}_{\sum_{j=0}^{k+1} N_j}]^T$ .

**Step 3:** Calculate the output weights  $\beta^{k+1}$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k * \mathbf{H}_{k+1}^T * (\mathbf{I} + \mathbf{H}_{k+1} * \mathbf{P}_k * \mathbf{H}_{k+1}^T)^{-1} * \mathbf{H}_{k+1} * \mathbf{P}_k \quad (10)$$

$$\beta^{(k+1)} = \beta^k + \mathbf{P}_{k+1} * \mathbf{H}_{k+1}^T * (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} * \beta^{(k)}) \quad (11)$$

**Step 4:** Set  $k = k + 1$ . Go to 2.

Algorithm 1: Online Sequential Extreme Learning Machine.

are calculated using the following equation (12).

$$ACC = \frac{TP + TN}{TP + FN + TN + FP} \quad (12)$$

where the values of true positive ( $TP$ ), true negative ( $TN$ ), false positive ( $FP$ ) and false negative ( $FN$ ). Equation (12) allows calculating the classification accuracy considering the number of total hits on the number of total samples. Although these equation (12) is defined for binary classification, the extension to multi-class classification implies only in (12) adding the correct answers for each class over the total samples of all classes.

To maximize ACC performance in the neural network trained by the OS-ELM, a estimation of optimal hyperparameter values is performed. The hyperparameters considered are the number of hidden neurons

( $L$ ) and  $C$  (regularization parameter). This estimation consists of performing simple cross-validations (training-validation), on a set of combinations of specific values for the two mentioned hyperparameters. In the case of  $L$ , these values range from 500 to 5,000 neurons in steps of 500 neurons. On the other hand, the value of the regularization parameter ranges from  $10^{-20}$  to  $10^{20}$ , in steps of 1 in the exponent. In this way, a wide search space of 410 points is approached for fingerprint classification problem (Zabala-Blanco et al., 2020b). Furthermore, indicate that a smaller portion of samples is used to estimate hyperparameters of the classifier, since it is computationally expensive to use the total number of samples in this process. For this estimation, a portion of samples will be considered such as those used by Zabala-Blanco et al.



Table 2: Details of the server used in the experiments.

Item	Description
CPU	Intel® Xeon® Gold 6140
RAM	128GB
GPU	Geforce GTX 1080ti
OS	Debian 10 64 bit

(2020b) in base ELM neural networks. Thus, 30,000 samples are used in total: 18,000 for training, 6,000 for validation, and 6,000 for testing.

### 3 RESULTS

#### 3.1 Software and Hardware

The OS-ELM algorithm was coded in C++ v11, using the CUDA v9.2, cuBLAS v9.2 and cuSOLVER v9.2 libraries (from the NVIDIA platform) to accelerate matrix operations such as multiplication and calculation of inverses. It should be noted that the specific operations of the OS-ELM algorithm that were brought to the GPU correspond to matrix multiplications, calculation of activation functions and calculation of pseudoinverse matrices. All experiments were carried out on the server detailed in Table 2.

#### 3.2 Experimental Results

It should be noted that in all the experiments carried out, the weights  $\mathbf{W}$  and biases  $\mathbf{b}$  between the input layer and the hidden layer of the neural network were random values generated by a uniform distribution in the intervals  $[-1, 1]$  and  $[0, 1]$  respectively Huang et al. (2006). Likewise, the activation function used in the hidden layer of the neural network corresponds to a sigmoid function defined as:  $g(x) = \frac{1}{1+e^{-x}}$ , given its universal approximation capability in a SLFN using algorithms based on ELM (Zabala-Blanco et al., 2020a).

##### 3.2.1 Hyperparameter Estimation

It can be seen in Figures 3, 4, and 5, that there is a concentration of high ACC within the range  $10^{-10}$  to  $10^{10}$  in  $C$ , where the central values in this range reach the ACC maximums. Likewise, there is a trend towards better ACC when the number of hidden neurons increases, especially in quantities greater than 3,000 neurons. However, in Figure 5 there are also good values between 1,500 and 2,500 neurons. According to the above, it can be established that those hyperparameter values that maximize ACC are within the mentioned regions.

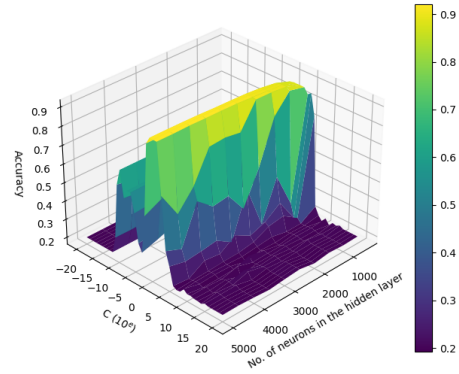


Figure 3: Accuracy for different hyperparameter values on Hong08 descriptor dataset.

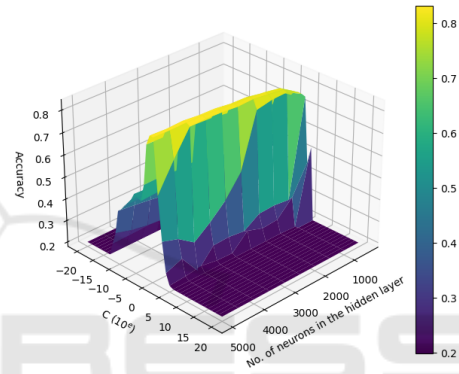


Figure 4: Accuracy for different hyperparameter values on Liu10 descriptor dataset.

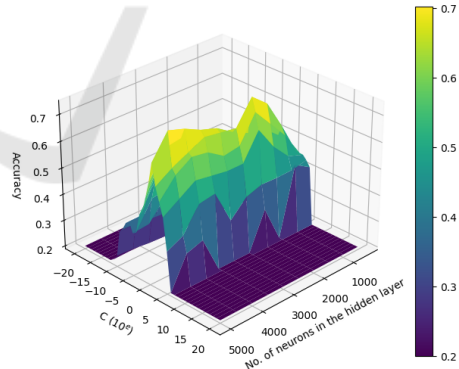


Figure 5: Accuracy for different hyperparameter values on Capelli02 descriptor dataset.

The hyperparameter values that maximize ACC in the neural network are presented in the following Table 3.

The optimal hyperparameters indicated in Table 3, were used in all experiments in Subsection 3.2.2.



Table 3: Optimal hyperparameter and maximum ACC obtained.

Dataset	Hyperparameter		ACC	
	$L$	$C$	Validation	Test
Hong08	4,500	10	0.92	0.92
Liu10	4,000	1	0.83	0.84
Capelli02	2,000	100	0.74	0.74

Table 4: Classification results on large fingerprint datasets.

Dataset	ACC		Time (sec)	
	Train	Test	Train	Test
Hong08	0.94	0.94	4,980.95	79.30
Liu10	0.83	0.83	4,655.93	64.38
Capelli02	0.75	0.75	2,638.78	47.14

### 3.2.2 Fingerprint Large Classification

An experiment was carried out with a large amount of data to evaluate the performance of the neural network trained by OS-ELM. This experiment consists of a training and testing on datasets with balanced fingerprint. For this purpose, 2,578,785 samples were used for each dataset, where each one is divided into two subsets, a training subset with 80% of the total samples (2,063,025 samples) and a test subset with 20 % of the total samples (515,760 samples). Table 4 shows the results.

Table 4 shows that regardless of the data set used, ACCs above 0.70 are achieved. However, it is observed that Hong08 and Liu10 show superiority in terms of ACC. On the other hand, when using Capelli02 the computational times are reduced almost by half compared to Hong08 and Liu10, mainly due to the smaller size of the hidden layer in the neural network. Although using Capelli02 requires less time than using Hong08, with the latter it improves by almost 0.2 of ACC, showing superiority in fingerprint classification.

In order to evaluate our proposal, the best results achieved were compared with other studies in the literature. Table 5 presents some measures of interest for comparative purposes.

It can be seen in Table 5 that our proposal is competitive in terms of testing ACC. Because the difference in ACC is not greater than 0.05 compared to other studies. Regarding training times, our proposal shows notable superiority since it achieves a learning speed almost 21 times greater than the fastest approach presented by Zabala-Blanco et al. (2020b). The learning speed is given by the quotient: no. of training samples/training time. Furthermore, it should be noted that our proposal allows us to deal with fingerprint classification for a large number of samples,

by processing almost 60 times more samples than the largest scenarios presented in the literature. Therefore, this approach is a suggestive alternative for practical application, since it is suitable to be introduced into large-scale person identification systems such as entry to mass events, entry into countries at airports, police control, among others.

## 4 CONCLUSIONS

In this study, we developed an approach to deal for fingerprint large classification problem. For this purpose, a single hidden layer feed-forward neural network trained by online sequential extreme learning machine was used. This approach was applied on three datasets with millions of descriptors (or samples) obtained from synthetic fingerprints. To achieve high classification accuracy, specialized fingerprint descriptors such as Hong08, Liu10 and Capelli02 were used. Regarding the results, our approach classified fingerprints with an accuracy of 0.94 when using Hong08 descriptors. Furthermore, to achieve this accuracy performance, less than 5,000 seconds of training time were required. This result is remarkable because a classifier based on neural networks was trained with millions of samples. Furthermore, this proposal shows a significant improvement in learning speed when compared to other approaches presented in the literature. These improvements are suggestive, since they allow us to address large-scale fingerprint classification when introduced into biometric systems intended for mass identification of people, which according to the authors' knowledge has not been addressed.

Finally, future work remains to address the identification of fingerprints with unbalanced classes, the use of deep features to improve performance in terms of accuracy, and extend the approach to process raw fingerprint images creating a fully automatic approach.

## REFERENCES

- Alias, N. A. and Radzi, N. H. M. (2016). Fingerprint classification using support vector machine. *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pages 105–108.
- An, L., Yang, S., and Bhanu, B. (2015). Efficient smile detection by extreme learning machine. *Neurocomputing*, 149:354–363.
- Borra, S. R., Reddy, G. J., and Reddy, E. S. (2018). Classification of fingerprint images with the aid of morpho-



Table 5: Comparison with results achieved by other studies.

Approach	Total samples	Testing ACC	Training time (sec)	Learning speed (samples/sec)
Hong08 + OS-ELM (Our proposal)	2,578,785	0.94	4,980.95	414.18
Hong08 + W-ELM2 (Zabala-Blanco et al., 2020b)	30,000	0.94	880.00	18.18
Bayesian deep CNN (Zia et al., 2019)	3,300	0.96	4,393.00	0.38
Novel CNN (Peralta et al., 2018)	40,000	0.99	960.00	10.42
CaffeNet (Peralta et al., 2018)	40,000	0.99	2,306.00	4.34
Pretrained VGG-S (Michelsanti et al., 2017)	3,300	0.96	108,000.00	0.03

- logical operation and agnn classifier. *Applied computing and informatics*, 14(2):166–176.
- Cao, K., Pang, L., Liang, J., and Tian, J. (2013). Fingerprint classification by a hierarchical classifier. *Pattern Recognition*, 46(12):3186–3197.
- Cao, W., Wang, X., Ming, Z., and Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing*, 275:278–287.
- Courrieu, P. (2008). Fast computation of moore-penrose inverse matrices. *arXiv preprint arXiv:0804.4809*.
- Deng, C., Han, Y., and Zhao, B. (2019). High-performance visual tracking with extreme learning machine framework. *IEEE transactions on cybernetics*, 50(6):2781–2792.
- Dorasamy, K., Webb, L., Tapamo, J., and Khanyile, N. P. (2015). Fingerprint classification using a simplified rule-set based on directional patterns and singularity features. In *2015 International Conference on Biometrics (ICB)*, pages 400–407. IEEE.
- El Hamdi, D., Elouedi, I., Fathallah, A., Nguyen, M. K., and Hamouda, A. (2018). Fingerprint classification using conic radon transform and convolutional neural networks. In *Advanced Concepts for Intelligent Vision Systems: 19th International Conference, ACIVS 2018, Poitiers, France, September 24–27, 2018, Proceedings 19*, pages 402–413. Springer.
- Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., García, S., Benítez, J. M., Pagola, M., Barrenechea, E., et al. (2015a). A survey of fingerprint classification part i: Taxonomies on feature extraction methods and learning models. *Knowledge-based systems*, 81:76–97.
- Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., García, S., Benítez, J. M., Pagola, M., Barrenechea, E., et al. (2015b). A survey of fingerprint classification part ii: Experimental analysis and ensemble proposal. *Knowledge-Based Systems*, 81:98–116.
- Galar, M., Sanz, J., Pagola, M., Bustince, H., and Herrera, F. (2014). A preliminary study on fingerprint classification using fuzzy rule-based classification systems. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 554–560. IEEE.
- Ge, S., Bai, C., Liu, Y., Liu, Y., and Zhao, T. (2017). Deep and discriminative feature learning for fingerprint classification. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1942–1946.
- Golub, G. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.
- Greville, T. (1959). The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations. *SIAM review*, 1(1):38–43.
- Guo, J.-M., Liu, Y.-F., Chang, J.-Y., and Lee, J.-D. (2014). Fingerprint classification based on decision tree from singular points and orientation field. *Expert Systems with Applications*, 41(2):752–764.
- Gupta, P. and Gupta, P. (2015). A robust singular point detection algorithm. *Applied Soft Computing*, 29:411–423.
- Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501.
- Huang, Q., Chang, S., Liu, C., Niu, B., Tang, M., and Zhou, Z. (2015). An evaluation of fake fingerprint databases utilizing svm classification. *Pattern Recognition Letters*, 60:1–7.
- Jung, H.-W. and Lee, J.-H. (2015). Noisy and incomplete fingerprint classification using local ridge distribution models. *Pattern recognition*, 48(2):473–484.
- Li, W., Chen, C., Su, H., and Du, Q. (2015). Local binary patterns and extreme learning machine for hyperspectral imagery classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(7):3681–3693.
- Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423.
- Liu, M. (2010). Fingerprint classification based on adaboost learning from singularity features. *Pattern Recognition*, 43(3):1062–1070.
- Lu, L., Zhang, X., and Xu, X. (2019). Hypercomplex extreme learning machine with its application in multispectral palmprint recognition. *PloS one*, 14(4):e0209083.
- Lu, S., Wang, X., Zhang, G., and Zhou, X. (2015). Effective algorithms of the moore-penrose inverse matrices for extreme learning machine. *Intelligent Data Analysis*, 19(4):743–760.
- Luo, J., Song, D., Xiu, C., Geng, S., Dong, T., et al. (2014). Fingerprint classification combining curvelet transform and gray-level cooccurrence matrix. *Mathematical Problems in Engineering*, 2014.



- Michelsanti, D., Ene, A.-D., Guichi, Y., Stef, R., Nasrollahi, K., and Moeslund, T. B. (2017). Fast fingerprint classification with deep neural networks. In *International Conference on Computer Vision Theory and Applications*, pages 202–209. SCITEPRESS Digital Library.
- Peralta, D., Triguero, I., García, S., Saeys, Y., Benitez, J. M., and Herrera, F. (2018). On the use of convolutional neural networks for robust classification of multiple fingerprint captures. *International Journal of Intelligent Systems*, 33(1):213–230.
- Rajanna, U., Erol, A., and Bebis, G. (2010). A comparative study on feature extraction for fingerprint classification and performance improvements using rank-level fusion. *Pattern Analysis and Applications*, 13:263–272.
- Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.
- Saeed, F., Hussain, M., and Aboalsamh, H. A. (2018). Classification of live scanned fingerprints using histogram of gradient descriptor. In *2018 21st Saudi computer society national computer conference (NCC)*, pages 1–5. IEEE.
- Saini, M. K., Saini, J., and Sharma, S. (2013). Moment based wavelet filter design for fingerprint classification. In *2013 International Conference on Signal Processing and Communication (ICSC)*, pages 267–270. IEEE.
- Shrein, J. M. (2017). Fingerprint classification using convolutional neural networks and ridge orientation images. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.
- Vitello, G., Sorbello, F., Migliore, G., Conti, V., and Vitabile, S. (2014). A novel technique for fingerprint classification based on fuzzy c-means and naive bayes classifier. In *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, pages 155–161. IEEE.
- Wang, R., Han, C., and Guo, T. (2016). A novel fingerprint classification method based on deep learning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 931–936. IEEE.
- Zabala-Blanco, D., Mora, M., Azurdia-Meza, C. A., Dehghan Firoozabadi, A., Palacios Játiva, P., and Soto, I. (2020a). Relaxation of the radio-frequency linewidth for coherent-optical orthogonal frequency-division multiplexing schemes by employing the improved extreme learning machine. *Symmetry*, 12(4):632.
- Zabala-Blanco, D., Mora, M., Barrientos, R. J., Hernández-García, R., and Naranjo-Torres, J. (2020b). Fingerprint classification through standard and weighted extreme learning machines. *applied sciences*, 10(12):4125.
- Zhang, Y., Wang, Y., Zhou, G., Jin, J., Wang, B., Wang, X., and Cichocki, A. (2018). Multi-kernel extreme learning machine for eeg classification in brain-computer interfaces. *Expert Systems with Applications*, 96:302–310.
- Zia, T., Ghafoor, M., Tariq, S. A., and Taj, I. A. (2019). Robust fingerprint classification with bayesian convolutional networks. *IET Image Processing*, 13(8):1280–1288.