v. 4, n. 27, 2024

Journal of
Engineering
Research

Acceptance date: 28/11/2024

MACHINE LEARNING FOR FACIAL RECOGNITION APPLIED TO LOCK-SINTELLIGENT

Victor de Rose Trunfo
Curso de Ciência da Computação, Grupo de Pesquisa em Inteligência Artificial Aplicada, Universidade do Extremo Sul Catarinense (UNESC), Criciúma - SC - Brasil

Leandro Neckel
Coorientador, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (UNESC), Criciúma - SC - Brasil

Merisandra Côrtes de Mattos
Orientadora, Curso de Ciência da Computação, Grupo de Pesquisa em Inteligência Artificial Aplicada, Universidade do Extremo Sul Catarinense (UNESC), Criciúma - SC - Brasil

Abstract: Security solutions for protecting property, such as conventional, smart and electronic locks, are available on the market. Some make use of biometric identification by means of fingerprints; however, facial recognition, using computer vision techniques through *deep learning*, has emerged as a promising alternative. The aim of this research is to analyze facial recognition algorithms applied to the problem of smart locks, using convolutional neural networks and evaluation metrics to measure the performance of the models generated. The nature of the research is applied and technology-based, using machine learning algorithms for facial recognition in smart locks. In terms of objectives, this is a descriptive study, as it evaluates the performance of the models generated by the methods and architectures employed using quality metrics in artificial neural networks. The results showed the promising effectiveness of the *DenseNet-121* and *DenseNet-169* architectures, combined with the *Fisherfaces* facial recognition method, in terms of precision, accuracy and F1, both in small-scale and large-scale analyses.
Keywords: Computer vision. *Deep learning*. Convolutional neural networks. *Eigenfaces*. *Fisherfaces*.

INTRODUCTION

Security is a constant concern in many sectors, including homes, commercial establishments and public institutions. Protecting property and guaranteeing access only to authorized persons are essential issues for promoting peace of mind and preventing unwanted incidents. In this context, conventional locks have been widely used, but they have limitations that can compromise their effectiveness.

Keyed locks, for example, are one of the most common solutions. However, the possibility of duplicating keys and the risk of them being lost or stolen pose a security threat. In addition, the process of manually unlocking and locking doors can be inconvenient and limiting, especially in shared environments where several people need access.

Faced with these limitations, alternatives have emerged such as electronic locks, which use technologies such as passwords or radio frequency identification. However, even these solutions are not without their flaws. Passwords can be discovered by malicious people, and RFID *tags* can be easily duplicated or stolen (AZNAR; PRADA, 2020). It is therefore necessary to look for more advanced and secure methods of access control.

One promising approach is the use of facial recognition, a biometric technique that makes it possible to identify people based on the unique characteristics of their faces (IGNACIO, 2021). Computer vision, artificial intelligence and machine learning play a key role in the development of efficient facial recognition algorithms and systems.

Facial recognition offers several advantages over other biometric solutions. By capturing a person's biometric measurements from a specific distance, without the need for physical interaction, it ensures greater convenience and speed of access (IGNACIO, 2021). In addition, facial recognition can identify people with criminal histories or legal problems, providing an additional level of security.

With the advance of technologies such as artificial intelligence and machine learning, computers not only perform repetitive tasks but also acquire learning capabilities (LUDERMIR, 2021). Deep learning has excelled in facial recognition, achieving results that are increasingly close to human capabilities (CHOI et al., 2020).

In this context, this research aims to analyze, by means of evaluation metrics, the models generated from machine learning algorithms for facial recognition applied to smart locks. To this end, two methods are used in this research, *Eigenfaces* and *Fisherfaces*, and five architectures are explored, including *ResNet-50*, *VGG16*, *VGG19*, *DenseNet-121* and *DenseNet-169*. Computer vision techniques, convolutional neural networks and evaluation metrics are applied to measure the performance of the facial recognition models generated.

The relevance of this study lies in the growing application of technologies such as artificial intelligence, machine learning and facial recognition, which can improve security and facilitate decision-making in various sectors. The use of smart locks with facial recognition has benefits such as efficiency, security and accessibility, contributing to a more fluid and secure experience for users (IGNACIO, 2021). In addition, solutions like this can be applied in different scenarios, such as homes, businesses and school environments, where access control is essential. Therefore, understanding and evaluating the performance of facial recognition models is fundamental to the development and improvement of these technologies.

RELATED WORKS

This section presents a review of related work on the subject of facial recognition applied to smart locks. The aim is to identify the existing contributions in the scientific literature, the approaches adopted, the limitations encountered and the research gaps not yet explored. These works serve as a basis for the development of this study, providing valuable *insights* and directions for research.

One of the first relevant works in the field of facial recognition was *Eigenfaces*, proposed by *Turk* and *Pentland* in 1991. This method uses the Principal Component Analysis (*PCA*) technique to extract the main facial features from a set of training images. Recognition is carried out by comparing these features with those of a test image. *Eigenfaces* obtained promising results, but had limitations in environments with variations in lighting and pose.

Belhumeur, Hespanha and Kriegman (1997) proposed *Fisherfaces* with the aim of overcoming the limitations of *Eigenfaces*. This method uses the Linear Discriminant Analysis (*LDA*) technique to extract discriminant characteristics that maximize the separation between classes. *Fisherfaces* proved to be more robust to variations in lighting and pose, providing a significant improvement in facial recognition performance.

Several studies have explored different facial recognition approaches applied to smart locks. Wang *et al.* (2016) proposed a facial recognition system based on *deep learning* using the *AlexNet* convolutional neural network architecture. The system was trained on a large set of facial images to recognize authorized individuals. The results showed a high success rate in facial recognition and demonstrated the effectiveness of using convolutional neural networks in this context.

Li *et al.* (2018) explored the use of convolutional neural networks for facial recognition in smart locks. The authors used the *VGG16* and *ResNet-50* architectures, trained on an extensive dataset. Comparative analyses were carried out between the architectures and parameters such as hit rate, processing time and robustness to variations in pose and lighting were evaluated. The results showed that both architectures performed well, with *ResNet-50* being slightly superior in terms of accuracy.

Lin and Wu (2020) used facial recognition to unlock vehicle doors, using principal component analysis to conduct the experiments and the *Eigenfaces* method. In training the model, since they used almost all people of Asian origin, the model becomes inaccurate if

used to recognize Westerners. They conclude that opening a door using easy recognition is more efficient and practical.

Despite significant advances in this area, there are still challenges to overcome. Some of these challenges include the need for robustness to variations in lighting, pose and occlusions, as well as dealing with privacy and security issues related to the storage and use of individuals' biometric information.

Given this panorama, this research aims to contribute to the advancement of the field of facial recognition applied to smart locks, exploring different architectures of convolutional neural networks and analyzing their performance in terms of accuracy, efficiency and robustness.

MATERIALS AND METHODS

The research approach is applied and technology-based, using machine learning algorithms for facial recognition in smart locks. In terms of objectives, this is a descriptive study, as it evaluates the performance of the models generated by the methods and architectures employed using quality metrics in artificial neural networks. As for the procedures, the research is bibliographical and experimental, comprising the manipulation of different neural network methods, parameters and architectures.

Figure 1 shows the general structure of this research, which included the stages of selecting and preparing the data set used, implementing the methods and architectures applied to facial recognition, testing and evaluating the models, and developing the facial recognition application for smart locks.

SELECTION AND PREPARATION OF THE DATA SET

The *VGGFace2* dataset was used as the basis for training, validating and testing the facial recognition models generated. *VGGFace2* consists of a large number of facial images collected from individuals of different ethnic origins, ages and genders. This dataset was selected because of its representativeness, diversity and wide variety of images of real people, including singers, actors, politicians, among others. The *VGGFace2* dataset, as well as the sources of information, image details and terms of use, is publicly available and can be accessed via the *VGGFace2 GitHub*[1] and the official website .[2]

Before using the dataset, the availability of different versions was checked and it was decided to use an *upscaled* version to ensure greater fidelity in the training. Preparation of the dataset included resizing, conversion to grayscale and other data preparation techniques.

RESIZING

Resizing is performed to adjust all images to a fixed size of 128x128 pixels using the cv2. resize function from the opencv2 library.

Before preparation, the images may have different sizes and vary in terms of resolution. Resizing ensures that all the images have the same dimension, which is necessary to feed the data into a machine learning model.

CONVERSION TO GRAYSCALE

This technique involves converting color images to grayscale. This is done to reduce the dimensionality of the data, since grayscale images have only one color channel, as opposed to the three channels (red, green and blue) of color images. Converting to grayscale simplifies image processing and can help reduce the amount of data needed to train the model.

1. Available at: https://github.com/ox-vgg/vgg_face2
Available at: https://www.robots.ox.ac.uk/~vgg/data/vgg_face2

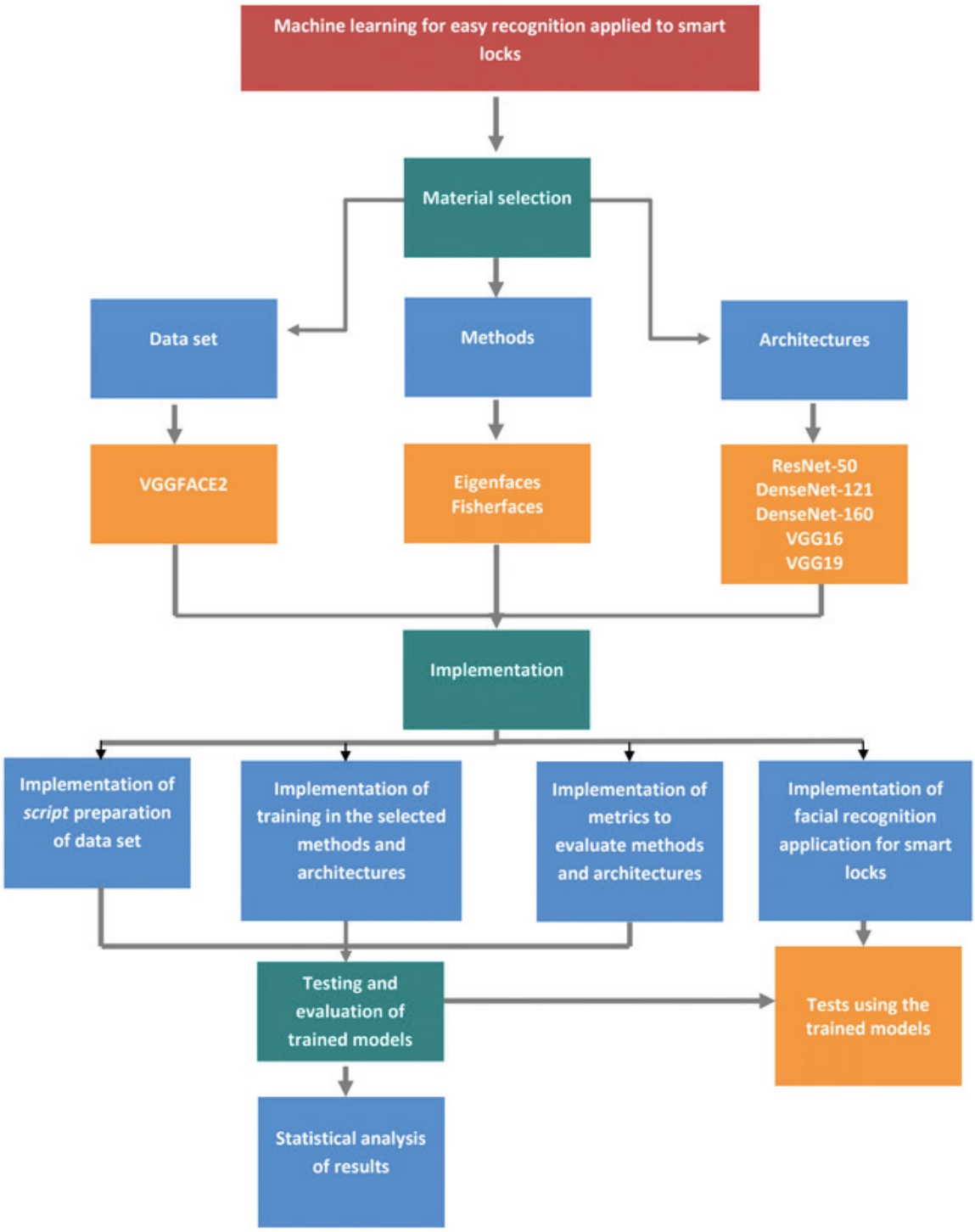

Figure 1 - General structure of the research

Source: From the author

DATA AUGMENTATION

Data augmentation is a technique used to increase the amount of training data by generating new samples based on existing samples. This is done by applying geometric transformations or modifying the properties of the images, such as rotation, mirroring, dimension, altering brightness/contrast and adding noise. According to the study by Kakadiaris *et al.* (2007), data augmentation is an effective strategy for diversifying the training set, thus improving the ability of the face recognition model to generalize and deal with different variations and input conditions.

FACE RECOGNITION APIS

A search of available *Application Programming Interfaces* (APIs) was carried out to find some solutions that used the *Eigenfaces* and *Fisherfaces* algorithms. These APIs were tested on demonstration bases in order to analyze their operation and accuracy in facial recognition. After these tests, it was decided to implement the facial recognition algorithms themselves based on *Eigenfaces* and *Fisherfaces*, as this would provide a better understanding of how they work. To do this, we reviewed the available literature and adjusted the parameters of the algorithms according to the requirements of the research.

The programming language used to code the *Eigenfaces* and *Fisherfaces* algorithms was *Python* version 3.10.9, using the *JupyterLab* development environment version 3.6.2. This programming language was chosen because it is widely used in machine learning and pattern recognition.

When configuring the development environment, some parameters were set as a *seed* for the value 42, which was selected at random, allowing the results to be replicated in future experiments. The video card's processing capacity was also used to speed up the processing of the images, which were imported directly from the computer used.

METHODS USED

Among the approaches used in facial recognition, two classic methods stand out, *Fisherfaces* and *Eigenfaces*, which were used in this research. Both are based on principal component analysis techniques and, according to Belhumeur *et al.* (1997), have been widely used in various facial recognition applications.

EIGENFACES

The *Eigenfaces* method is based on principal component analysis for facial recognition, capturing the main components of variation present in the training images, known as "eigenfaces". These eigenfaces represent the most significant variations found in the faces of the training set Pentland, A. (1991).

During the training process of the *Eigenfaces* method, the images are transformed into a low-dimensional space and represented as linear combinations of the eigenfaces. During the test phase, the face of a new image is projected into this subspace and compared with the projections of the training faces in order to determine the class of belonging according to the article by Belhumeur *et al.* (1997).

According to Pentland, A. (1991) the *Eigenfaces* model offers a computationally efficient approach to face recognition, being able to deal with a large number of images and individuals. However, it can be sensitive to variations in lighting and facial rotation, which can affect its accuracy in certain scenarios.

FISHERFACES

The *Fisherfaces* method, also known as *Fisher's Linear Discriminant*, is a dimensionality reduction technique that seeks to maximize separability between different image classes. The aim of this method is to find the most discriminating characteristics in the subspaces, with the greatest interclass variability and the least intraclass variability according to BELHUMEUR *et al.* (1997).

During the training of the *Fisherfaces* model, the images are projected onto a low--dimensional feature space in which the discriminant information is maximized. These extracted features are used to classify new images during the testing phase. *Fisherfaces* offers a robust and efficient approach to face recognition and is particularly suitable for problems with a limited number of samples per class, as explained in the article by Yang *et al.* (2011).

ARCHITECTURES USED

In this facial recognition study, a set of convolutional neural network architectures was used for training and evaluating the methods. The architectures used were carefully selected, taking into account their distinct characteristics and their performance in computer vision tasks, according to the scientific literature in the area. Five architectures were implemented in this research: *ResNet-50*, *VGG16*, VGG-19, *DenseNet-121* and *DenseNet-169*.

ResNet-50, according to He *et al.* (2016), is recognized for its ability to train deep neural networks, overcoming the challenge known as performance degradation[3] which is common when the number of layers in the network increases. This architecture employs the concept of residual connections, allowing information to flow more easily through the layers and promoting more effective learning.

The *VGG16* architecture according to Simonyan *et al.* (2014) uses convolutional filters of reduced size (3x3) in several consecutive layers, followed by *pooling* layers. This approach allows the extraction of detailed features from images and results in a model suitable for face recognition.

VGG19, also according to Simonyan *et al.* (2014), is a variation of *VGG16* that expands the network's representation capacity by adding convolutional and *pooling* layers. With a larger number of parameters, *VGG19* is able to capture more complex and subtle features in images. However, it is important to note that the increase in complexity may imply a longer training time.

According to Huang *et al.* (2017), the *DenseNet-121* architecture has dense connections between the layers, which allows for a more direct flow of information. Each layer receives information from the previous layers, promoting better communication between them. This improves the use of features at different levels of abstraction, resulting in a compact and efficient model.

DenseNet-169, also according to Huang *et al.* (2017), is an extension of the previous architecture with a greater number of layers and the ability to learn more complex representations and capture subtle nuances in images. However, it is important to consider that *DenseNet-169* may require more intensive computational resources due to its greater complexity.

MODEL TESTING AND EVALUATION

To carry out the experiments and train the facial recognition models, we used a computer with the following specifications: AMD Ryzen 5 5600X processor, 32GB of RAM with a frequency of 3600MHz, RTX 3070 Ti video card with 8GB capacity, 2TB NVME SSD storage and *Windows* 11 64-bit operating system.

The facial recognition models generated by the different methods and architectures had their performance evaluated in this research. To do this, tests were carried out on two scales, small and large. The small scale involved 25 people, with 500 images available for training and validation, and 125 images for testing. The large scale involved 100 people, with 2000 images for training and validation, and 500 images for testing.

3. According to Smith (2018), performance degradation occurs when there is a reduction in the capacity or efficiency of a system in relation to a previous state or a reference system. This degradation can be caused by various factors, such as changes in environmental conditions, wear and tear on components or inadequate algorithms.

In order to obtain scientifically relevant results, it was necessary to standardize the training data in order to provide the same test protocol and learning capacity for the different models and architectures. Thus, taking into account their particularities, these standards were established after studying available scientific material on the use of facial recognition methods and carrying out an extensive number of previous tests, approximately 400. These tests took into account the diversity of variables, the number of components (*eigenfaces* and *fisherfaces*), the form of standardization, their proportion, as well as the variation in the number of people (10 to 500), using different numbers of images per person.

In this way, a test pattern was established for all the methods and architectures used in this research, which comprised:

a) **Training bases**: After analyzing the results of various tests to decide the proportion, 80% of the images of people were used, as suggested in the book "Deep Learning" (Goodfellow, Bengio & Courville, 2016), which were randomly selected by a script developed in Python. These were 500 images for the small-scale tests and 2000 images for the large-scale tests. Of these, 20% of the images were used in a validation set, which provides greater reliability when training. For the small-scale tests, 400 images were used for training and 100 for validation. For the large-scale tests, 1600 images were used for training and 400 for validation;

b) **Test bases**: the remaining 20% of the selected images were also randomly selected to make up the test set, using 125 images for the small-scale tests and 500 images for the large-scale tests.

The division of data used makes it possible to evaluate the performance of the models on different sets of training, validation and test data, covering both a small scale and a larger scale.

During the model training and evaluation phases, 200 training runs were carried out, 10 of which were for each of the five architectures (*ResNet-50*, *VGG16*, VGG-19, *DenseNet-121* and *DenseNet-169*), giving a total of 50 for each of the methods used (*eigenfaces* and *fisherfaces*) and 100 for each of the scales (small and large).

The models were trained using the Adam optimizer and a function that allowed training to be carried out until the loss value, based on validation, did not obtain better results for a period of 30 epochs. The definition of this measure took into account the variability of the architectures, with some taking longer than others to obtain significant results, and the number of epochs per training session was also taken into account when analyzing the results.

The performance of the models generated by the machine learning methods and architectures used in this research was analyzed using the accuracy metrics[4] , precision, *recall* and *F1-score*[5] . These metrics provide a comprehensive assessment of the facial recognition capacity of the models developed.

4. Following the definition proposed by Hastie et al. (2009) as the measure that indicates the proportion of correct classifications in relation to the total number of examples evaluated

5. According to Powers (2011), precision, *recall* and *F1-score* are metrics widely used in the evaluation of classification systems. Precision is defined as the proportion of examples correctly classified as positive in relation to the total number of examples classified as positive, while *recall* is the proportion of positive examples correctly classified in relation to the total number of positive examples. The *F1-score* is a measure that combines precision and *recall* into a single metric, and is calculated as the harmonic mean between these two measures

APPLYING FACIAL RECOGNITION TO ELECTRONIC LOCKS

The research also included the development of an application for facial recognition in electronic locks. To this end, a facial authentication system was implemented using convolutional neural networks combined with the *Eigenfaces* and *Fisherfaces* methods, which were applied to reduce the dimensionality of the data and improve the efficiency of the authentication system.

The system uses a webcam to capture frames in real time, detects faces in the frames using the OpenCV face classifier, and performs authentication based on the region of the face detected.

The convolutional neural network architectures were trained using a set of images containing the faces of different individuals, as explained earlier in this article. During training, the model learned to extract the relevant characteristics for identifying and authenticating individuals.

The application pre-processes the captured images, resizing them to the size expected by the model and normalizing the *pixel* values to the range between 0 and 1. It then applies the trained model, along with the *Eigenfaces* or *Fisherfaces* methods, to predict the class corresponding to the face present in the region of interest. Subsequently, the name of the predicted class is displayed above the rectangle surrounding the face detected in the *webcam* frame.

The application also calculates and displays the average accuracy of the last *ticks*, which represents the average of the authentication probabilities obtained in the last frames processed, which provides an estimate of the reliability of the authentication system.

RESULTS AND DISCUSSION

This study used statistical methods to evaluate the models generated, which were carried out using IBM SPSS software version 21 in its demonstration version. The statistical analysis initially included the Shapiro-Wilk normality test to check whether the data relating to the metrics of precision, accuracy, *F1-score* and total processing time had a normal probability distribution or not, considering the significance level ($\alpha = 5\% = 0.05$) and the 95% confidence interval.

The Shapiro-Wilk normality test showed that the precision (p-value < 0.001), accuracy (p-value < 0.001), *F1-score* (p-value < 0.001) and total time (p-value < 0.001) data did not show a normal probability distribution, as the p-value was less than 0.05 (α).

Therefore, the test used to verify whether there is a significant difference between the values of precision, accuracy, *F1-score* and total processing time presented in the experiments carried out was the Kruskal-Wallis non-parametric H-test.

The Kruskal-Wallis test is a non-parametric test used to compare the means of three or more independent groups. In this study, it was used to compare the means of the face recognition architectures in relation to the metrics of precision, accuracy, *F1-score* and total processing time.

In the case of architectures with equal letters, this means that the averages of the corresponding groups are statistically equal. For example, when we mention that the average accuracy of the *VGG19* (*LDA*) architecture was statistically equal to the average accuracies of the architectures with letters *a* and *b*, this indicates that there is no statistically significant difference between these architectures in terms of average accuracy. This comparison was carried out using the Kruskal-Wallis test, which identifies statistically significant differences between the means of the groups.

ANALYSIS OF SMALL-SCALE TESTS

In the analysis of the small-scale tests, the metrics of precision, accuracy, *F1-score* and total processing time of the *DenseNet-121*, *DenseNet-169*, *VGG16*, *VGG19* and *ResNet-50* face recognition architectures were considered, using the *Eigenfaces* (treated as *PCA*) and *Fisherfaces* (treated as *LDA*) models; the number n represents the number of models trained for the tests.

The *DenseNet-121* (*LDA*) and *DenseNet-169* (*LDA*) architectures had the highest accuracy averages, with values of 0.952 and 0.951, respectively. These averages were statistically equivalent, indicating consistent performance between the two architectures. The *VGG19* (*LDA*) architecture also obtained promising results, although it showed greater variability in the results shown in Table 1.

In the accuracy metric, the *DenseNet-121* (*LDA*) and *DenseNet-169* (*LDA*) architectures also obtained the highest averages, with statistically equivalent results. Once again, the *VGG19* (*LDA*) architecture showed the greatest variability in the results shown in Table 2.

For the *F1-score* metric, the *DenseNet-121* (*LDA*) and *DenseNet-169* (*LDA*) architectures stood out with the highest averages, while the *VGG19* (*LDA*) architecture showed variable results as shown in Table 3.

Analysis of the total processing time of the facial recognition architectures on the small scale revealed significant variations, with the *VGG16* (*PCA*) and *VGG19* (*PCA*) architectures having the lowest average processing times, with values of 14.200 minutes and 4.447 minutes, respectively. However, these architectures also had a relatively high standard deviation, indicating greater variability in processing times between runs, as shown in Table 4.

ANALYSIS OF LARGE-SCALE TESTS

In the analysis of large-scale tests, the metrics of precision, accuracy, *F1-score* and total processing time of the *DenseNet-121*, *DenseNet-169*, *VGG16*, *VGG19* and *ResNet-50* facial recognition architectures were also considered, using the eigenfaces (*PCA*) and fisherfaces (*LDA*) models.

For the accuracy metric, the *DenseNet-121* (*LDA*), *ResNet-50* (*LDA*) and *DenseNet-169* (*LDA*) architectures obtained the highest averages, with values of 0.973, 0.961 and 0.954, respectively. These averages were statistically equivalent, indicating consistent performance between these architectures. On the other hand, the *VGG16* (*LDA*) and *VGG19* (*LDA*) architectures had significantly lower averages and greater variability in the results shown in Table 5.

When analyzing accuracy, it was observed that the *DenseNet-121* (*LDA*), *ResNet-50* (*LDA*) and *DenseNet-169* (*LDA*) architectures also had the highest averages, with statistically equivalent results between them. The *DenseNet-121* (*PCA*), *VGG19* (*LDA*) and *DenseNet-169* (*PCA*) architectures had slightly lower averages, while the *VGG16* (*LDA*), *ResNet-50* (*PCA*), *VGG16* (*PCA*) and *VGG19* (*PCA*) architectures had even lower averages, as shown in Table 6.

With regard to the *F1-score* metric, the *DenseNet-121* (*LDA*), *ResNet-50* (*LDA*) and *DenseNet-169* (*LDA*) architectures stood out with the highest averages, while the *VGG19* (*LDA*) and *VGG16* (*LDA*) architectures showed lower averages and greater variability in the results. The *DenseNet-121* (*PCA*) and *ResNet-50* (*PCA*) architectures also showed lower averages compared to the other architectures shown in Table 7.

Analysis of the total processing time revealed that the *ResNet-50* (*PCA*), *ResNet-50* (*LDA*), *DenseNet-169* (*LDA*), *DenseNet-121* (*LDA*), *DenseNet-121* (*PCA*) and *DenseNet-169* (*PCA*) architectures had relatively high average pro-

Precision	n	Average	Standard deviation	Minimum	Maximum
DenseNet-121 (LDA)be	10	0,952	0,026	0,903	0,982
DenseNet-169 (LDA)be	10	0,951	0,024	0,913	0,987
VGG19 (LDA)bde	10	0,928	0,066	0,788	1,000
ResNet-50 (LDA)bce	10	0,914	0,044	0,846	0,959
VGG16 (LDA)bce	10	0,902	0,056	0,802	0,962
DenseNet-121 (PCA)ae	10	0,806	0,041	0,750	0,904
DenseNet-169 (PCA)acd	10	0,776	0,044	0,713	0,857
ResNet-50 (PCA)ac	10	0,701	0,074	0,536	0,785
VGG16 (PCA)a	10	0,433	0,135	0,080	0,565
VGG19 (PCA)a	10	0,117	0,128	0,025	0,461

Table 1. Statistical measures for precision in small-scale tests

Source: From the author.

Acuraria	n	Average	Standard deviation	Minimum	Maximum
DenseNet-169 (LDA)be	10	0,938	0,028	0,896	0,984
DenseNet-121 (LDA)be	10	0,937	0,035	0,872	0,976
VGG19 (LDA)bde	10	0,901	0,101	0,672	1,000
ResNet-50 (LDA)bce	10	0,890	0,063	0,776	0,952
VGG16 (LDA)bce	10	0,772	0,276	0,240	0,952
DenseNet-121 (PCA)ae	10	0,757	0,071	0,616	0,856
DenseNet-169 (PCA)acd	10	0,727	0,045	0,664	0,816
ResNet-50 (PCA)ac	10	0,642	0,078	0,520	0,744
VGG16 (PCA)a	10	0,423	0,125	0,104	0,568
VGG19 (PCA)a	10	0,139	0,105	0,072	0,432

Table 2. Statistical measures for accuracy in small-scale tests

Source: From the author.

F1-score	n	Média	Desvio padrão	Mínimo	Máximo
DenseNet-169 (LDA)be	10	0,939	0,028	0,898	0,984
DenseNet-121 (LDA)be	10	0,938	0,034	0,877	0,976
VGG19 (LDA)bde	10	0,905	0,095	0,691	1,000
ResNet-50 (LDA)bce	10	0,891	0,061	0,783	0,951
VGG16 (LDA)bce	10	0,790	0,242	0,318	0,953
DenseNet-121 (PCA)ae	10	0,759	0,067	0,636	0,861
DenseNet-169 (PCA)acd	10	0,723	0,048	0,657	0,816
ResNet-50 (PCA)ac	10	0,636	0,084	0,481	0,746
VGG16 (PCA)a	10	0,399	0,130	0,066	0,548
VGG19 (PCA)a	10	0,097	0,115	0,031	0,419

Table 3 Statistical measures for *F1-score* in small-scale tests

Source: From the author.

F1-score	n	Average	Standard deviation	Minimum	Maximum
DenseNet-169 (LDA)be	10	0,939	0,028	0,898	0,984
DenseNet-121 (LDA)be	10	0,938	0,034	0,877	0,976
VGG19 (LDA)bde	10	0,905	0,095	0,691	1,000
ResNet-50 (LDA)bce	10	0,891	0,061	0,783	0,951
VGG16 (LDA)bce	10	0,790	0,242	0,318	0,953
DenseNet-121 (PCA)ae	10	0,759	0,067	0,636	0,861
DenseNet-169 (PCA)acd	10	0,723	0,048	0,657	0,816
ResNet-50 (PCA)ac	10	0,636	0,084	0,481	0,746
VGG16 (PCA)a	10	0,399	0,130	0,066	0,548
VGG19 (PCA)a	10	0,097	0,115	0,031	0,419

Table 4: Statistical measures for total processing time in small-scale tests

Source: From the author.

Precision	n	Average	Standard deviation	Minimum	Maximum
DenseNet-121 (LDA)a	10	0,973	0,008	0,956	0,983
ResNet-50 (LDA)a	10	0,961	0,016	0,931	0,981
DenseNet-169 (LDA)a	10	0,954	0,014	0,928	0,981
VGG16 (LDA)ab	10	0,767	0,323	0,051	1,000
VGG19 (LDA)ab	10	0,766	0,332	0,140	1,000
DenseNet-121 (PCA)b	10	0,748	0,039	0,696	0,824
DenseNet-169 (PCA)b	10	0,676	0,066	0,589	0,776
ResNet-50 (PCA)b	10	0,657	0,032	0,605	0,698
VGG16 (PCA)c	10	0,061	0,141	0,005	0,463
VGG19 (PCA)c	10	0,013	0,010	0,000	0,031

Table 5: Statistical measures for accuracy in large-scale tests

Source: From the author.

Acurária	n	Average	Standard deviation	Minimum	Maximum
DenseNet-121 (LDA)a	10	0,964	0,011	0,938	0,980
ResNet-50 (LDA)a	10	0,946	0,025	0,904	0,976
DenseNet-169 (LDA)a	10	0,939	0,023	0,892	0,974
DenseNet-121 (PCA)b	10	0,659	0,050	0,602	0,754
VGG19 (LDA)ab	10	0,621	0,397	0,040	1,000
DenseNet-169 (PCA)b	10	0,592	0,070	0,504	0,700
VGG16 (LDA)ab	10	0,583	0,353	0,020	1,000
ResNet-50 (PCA)b	10	0,572	0,046	0,492	0,636
VGG16 (PCA)c	10	0,067	0,109	0,012	0,374
VGG19 (PCA)c	10	0,034	0,015	0,010	0,060

Table 6: Statistical measures for accuracy in large-scale tests

Source: From the author.

F1-score	n	Average	Standard deviation	Minimum	Maximum
DenseNet-121 (LDA)a	10	0,965	0,011	0,939	0,980
ResNet-50 (LDA)a	10	0,947	0,024	0,907	0,976
DenseNet-169 (LDA)a	10	0,940	0,021	0,897	0,974
DenseNet-121 (PCA)b	10	0,665	0,047	0,607	0,758
VGG19 (LDA)ab	10	0,655	0,389	0,049	1,000
VGG16 (LDA)ab	10	0,628	0,345	0,018	1,000
DenseNet-169 (PCA)b	10	0,594	0,075	0,501	0,711
ResNet-50 (PCA)b	10	0,575	0,045	0,505	0,636
VGG16 (PCA)c	10	0,049	0,111	0,004	0,366
VGG19 (PCA)c	10	0,013	0,007	0,001	0,024

Table 7. Statistical measures for *F1-score* in large-scale tests

Source: From the author.

Total time (in minutes)	n	Average	Standard deviation	Minimum	Maximum
ResNet-50 (PCA)b	10	38,625	9,395	26,250	55,000
ResNet-50 (LDA)b	10	35,958	10,369	21,250	53,333
DenseNet-169 (LDA)b	10	35,250	11,938	24,167	64,583
DenseNet-121 (LDA)b	10	34,167	8,020	25,833	50,833
DenseNet-121 (PCA)b	10	32,792	4,222	27,917	39,583
DenseNet-169 (PCA)b	10	32,292	3,885	26,250	37,500
VGG16 (PCA)a	10	19,792	20,585	12,917	78,333
VGG19 (PCA)a	10	13,583	1,392	12,917	17,083
VGG19 (LDA)a	10	13,583	1,524	12,917	17,500
VGG16 (LDA)a	10	13,500	1,061	12,917	15,833

Table 8. Statistical measures for *F1-score* in large-scale tests

Source: From the author.

cessing times. On the other hand, the *VGG16* (*PCA*) and *VGG19* (*PCA*) architectures had the lowest average processing times, with values of 19.792 minutes and 13.583 minutes, respectively. However, these architectures also showed a relatively high standard deviation, indicating greater variability in processing times between runs, as shown in Table 8.

The Kruskal-Wallis test showed that the *VGG16* (*LDA*) and *VGG19* (*LDA*) architectures had statistically equivalent averages to *ResNet-50*, *DenseNet-121* and *DenseNet-169* in terms of precision, accuracy and *F1-score*. However, the variability observed in these results suggests a sensitivity of these architectures to different data conditions and characteristics.

DISCUSSION OF RESULTS

Although the studies mentioned did not provide specific figures for metrics such as precision, accuracy and F1-score, it is important to note that all the studies reported satisfactory performance in facial recognition in smart locks.

In the study by Wang et al. (2016), the system based on the AlexNet architecture achieved a high hit rate in facial recognition. Although precise numerical values were not mentioned, the results indicated promising performance.

Li et al. (2018) carried out comparative analyses between the VGG16 and ResNet-50 architectures. Although exact metric values

were not provided, the results showed that both architectures performed well, with ResNet-50 being slightly superior in terms of accuracy.

In this research, the DenseNet-121 (LDA) and DenseNet-169 (LDA) architectures also showed consistent performance in terms of precision, accuracy and F1-score. Although we don't have specific figures for these metrics, the results were considered satisfactory.

Therefore, although the studies mentioned did not provide specific figures, it is possible to conclude that they all showed promising performance in facial recognition in smart locks. Convolutional neural network architectures, including AlexNet, VGG16, ResNet-50, DenseNet-121 and DenseNet-169, have proven effective for this purpose, offering consistent results in terms of precision, accuracy and F1-score.

However, it is important to consider the differences in the experimental configurations, the data sets used and the metrics evaluated, which may influence the results obtained. Each study is unique in its settings and objectives, and therefore a direct comparison between numerical results may not be possible.

In this way, the results of this research corroborate the related studies mentioned above, highlighting the effectiveness of using convolutional neural networks in facial recognition in smart locks. Furthermore, it is essential to consider that there are still challenges to be overcome, such as robustness to variations in lighting, pose and occlusions, as well as issues related to the privacy and security of biometric data.The ongoing research mentioned at the end highlights the intention to contribute to the advancement of the field by exploring different convolutional neural network architectures and analyzing their performance in terms of accuracy, efficiency and robustness. This type of study is essential for continually improving facial recognition technologies applied to smart locks.

CONCLUSION

Based on the results obtained, the *DenseNet-121* (*LDA*) and *DenseNet-169* (*LDA*) architectures showed promising performance in relation to the metrics of precision, accuracy and *F1-score*, both in small-scale and large--scale analysis. These architectures stood out compared to the others evaluated, indicating that they are good options for facial recognition in smart locks.

However, it is important to consider the variability observed in the *VGG19* (*LDA*) and *VGG16* (*LDA*) architectures, which showed more unstable results, and that this study sought to use a real database to obtain credible results in scenarios with high variability in facial position and lighting level. By adopting this approach with images of real people, the study aimed to accurately reflect the conditions faced in biometric authentication based on facial recognition in real-world environments, with a focus on application in smart locks.

It is important to note that this study has limitations, and small-scale analysis may not fully represent the performance of architectures in large-scale scenarios. It is recommended that future studies be conducted with a larger number of samples and images from different individuals to validate and strengthen the conclusions reached in this research.

This study can make a significant contribution to the field of facial recognition by offering an analysis of the methods and architectures contemplated for biometric authentication based on facial recognition, particularly in the context of smart locks. By considering methods such as *Eigenfaces* and *Fisherfaces*, and different convolutional neural network architectures, it was shown that the *DenseNet-121* (*LDA*) and *DenseNet-169* (*LDA*) architectures presented a satisfactory balance between performance and stability. However, it is essential to continue with additional studies to investigate other relevant aspects and metrics, as well as carrying out analyses on an even larger scale.

REFERENCES

ABDELMINAAM, Diaa Salama *et al.* **A deep facial recognition system using computational intelligent algorithms**. PLoS ONE, [s. l.], v. 15, n. 12 December, p. 1–27, 2020. Disponível em: http://dx.doi.org/10.1371/journal.pone.0242269.

BELHUMEUR, P. N.; HESPANHA, J. P.; KRIEGMAN, D. J. ***Eigenfaces* vs. *Fisherfaces*: Recognition Using Class Specific Linear Projection**. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 19, n. 7, p. 711-720, jul. 1997. https://doi.org/10.1109/34.598228.

CHELALI, F. Z.; DJERADI, A.; DJERADI, R. **Linear discriminant analysis for face recognition**. In: INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING AND SYSTEMS, Ouarzazate, Morocco, p. 1-10, 2009. https://doi.org/10.1109/MMCS.2009.5256630.

CHOI, Rene Y. *et al.* **Introduction to machine learning, neural networks, and deep learning**. Translational Vision Science and Technology, [s. l.], v. 9, n. 2, p. 1–12, 2020.

DENG, Li *et al.* **Deep Learning : Methods and Applications**. [s. l.], v. 7, n. 2013, p. 197–387, 2014.

GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. (2016). **Deep Learning**. MIT Press.

HASTIE, T., TIBSHRANI, R., & FRIEDMAN, J. (2009). **The Elements of Statistical Learning**. Springer Science & Business Media.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. **Deep residual learning for image recognition**. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

HUANG, G., LIU, Z., VAN DER MAATEN, L., & WEINBERGER, K. Q. (2017). **Densely connected convolutional networks**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

KAKADIARIS, I. A., PASSALIS, G., TODERICI, G., MURTUZA, N., & LU, Y. (2007). **Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(4), 640-649.

LI, Z., ZHANG, Y., & FU, Y. (2018). **Face recognition unlocking uses principal component analysis to control the vehicle door system**. 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 1-4.

POWERS, D. M. (2011). **Evaluation: From precision, *recall* and F-measure to ROC, informedness, markedness & correlation**. Journal of Machine Learning Technologies, 2(1), 37-63.

SIMONYAN, K., & ZISSERMAN, A. (2014). **Very deep convolutional networks for large-scale image recognition**. arXiv preprint arXiv:1409.1556.

SMITH, J. **Degradação de Desempenho em Sistemas de Reconhecimento Facial**. Journal of Computer Vision and Pattern Recognition, vol. 20, no. 3, pp. 123-145, 2018

YANG, M., Zhang, L., Zhang, D., & Li, X. (2011). **Fisher discrimination dictionary learning for sparse representation**. IEEE Transactions on Image Processing, 20(4), 929-941.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., & SIMONCELLI, E. P. (2016). **Image quality assessment: From error visibility to structural similarity.** IEEE Transactions on Image Processing, 13(4), 600-612.