Role of Distance Measures in Approximate String Matching Algorithms for Face Recognition System

B. Krishnaveni(✉) and S. Sridhar

Information Science and Technology Department, Anna University, Chennai, India
krishnaveni1116@gmail.com

Abstract. This paper is based on the recognition of faces using string matching. The approximate string matching is a method for finding an approximate match of a pattern within a string. Exact matching is impracticable for a larger amount of data as it involves more time. Those issues can be solved by finding an approximate match rather than an exact match. This paper aims to experiment with the performance of approximation string matching approaches using various distance measures such as Edit distance, Longest Common Subsequence (LCSS), Hamming distance, Jaro distance, and Jaro-Winkler distance. The algorithms generate a near-optimal solution to face recognition system with reduced computational complexity. This paper deals with the conversion of face images into strings, matching those image strings by using the approximation string matching algorithm that determines the distance and classifies a face image based on the minimum distance. Experiments have been performed with FEI and ORL face databases for the evaluation of approximation string matching algorithms and the results demonstrate the utility of distance measures for the face recognition system.

Keywords: Approximate · String matching · Face recognition · Edit distance · LCSS · Hamming · Jaro distance

1 Introduction

Face recognition is the identification or verification of a person's face from a database of different person's faces [1]. Nowadays, face recognition has become a widely used application in mobile phones, and robotics, and security systems. It has also received attention in applications such as surveillance, human-computer interaction, access control, a criminal investigation, border control, and smart cars. Certain applications such as text searching, DNA subsequence searching, and signal processing and object recognition require faster and near-optimal solutions. Face recognition is also one such application that may require a near-optimal and faster solution.

Several face recognition approaches deal with controlled face recognition [2–8], but only a few approaches address the face recognition under uncontrolled conditions include partial occlusion, pose changes, illumination changes, and expression changes

Published by Springer Nature Switzerland AG 2020
A. Chandrabose et al. (Eds.): ICCIDS 2020, IFIP AICT 578, pp. 157–169, 2020.
https://doi.org/10.1007/978-3-030-63467-4_12

[9–11]. Uncontrollable face recognition is a difficult and challenging task. An efficient face recognition system should address those conditions.

String matching [12] is a high level structural and syntactic technique for finding similarity between two strings. In other words, a string matching algorithm finds the location of a certain pattern in a larger amount of text or string. The applications of string matching include search engines, object recognition, speech recognition, Information retrieval systems, and molecular biology.

An approximation algorithm [13] is designed to solve optimization problems in polynomial time. A method based on approximation yields a near-optimal solution with a guarantee on the solution's quality. An approximation algorithm for string matching [14, 15] is designed for determining strings that match a pattern approximately instead of exactly.

In this paper, the face recognition problem is modeled as the optimization problem since the approximation algorithm tries to find the distances between the test face image and all the database face images and select the minimum of those distances. Thus, the algorithm performs image matching based on the minimum distance obtained which becomes a minimization problem and any minimization or maximization problem is considered as the optimization problem. The overall goal of the work presented in this paper is to handle the challenges in the uncontrolled face recognition with different distance measures using approximation matching algorithms.

The rest of the paper is ordered as follows. Section 2 presents literature work on face recognition techniques, the methodology is described in Sect. 3, the experimental results are discussed in Sect. 4, and the last section gives the conclusion of the paper.

2 Related Work

Conventional methods for face recognition such as PCA (principal component analysis) [2, 3], LDA (linear discriminant analysis) [4, 5], ICA (independent component analysis) [6], and HMM [7] perform face recognition under controlled conditions. These methods do not perform well when there are uncontrolled conditions such as lighting, pose, expression variations, and occlusion.

The different approaches that deal with the face recognition under uncontrolled conditions include SRC (Sparse representation-based classification) [20], SRC-MRF (SRC with Markov random fields) [21], HQM (half quadratic with multiplicative form) [22], Virtual samples and SRC based algorithm [23] and GSR (group sparse representation-based) [24]. The computational cost of these sparse representation based approaches is more and require a larger number of training samples. Local feature matching face recognition methods such as metric learned extended robust point set matching [25], dynamic image-to-classwarping for face recognition [26] handles uncontrolled face recognition using local features such as image patches.

The research towards face recognition using string matching is very less. One of the related work is an ensemble string matching [27], performs string-to-string matching for the recognition of faces. The human face profile [28] is another work based on representing a face as a sequence of strings.

The latest research face recognition systems based on deep learning [29–33] are succeeded for controlled face recognition, but not producing successful results in uncontrolled conditions. The algorithms presented in this paper make use of different string distance measures for performing face recognition and yield a good recognition rate in case of face recognition application.

3 Methodology

3.1 Face Recognition System Model

Figure 1 demonstrates the architecture of the face recognition system. The various phases in the proposed system are feature extraction, clustering, generation of strings, matching test and database faces. In the feature extraction phase, small patches of m × n pixels are extracted from database gallery images of size p × q pixels. Patches are converted into patch vectors. Here, patch vectors are considered as feature vectors.

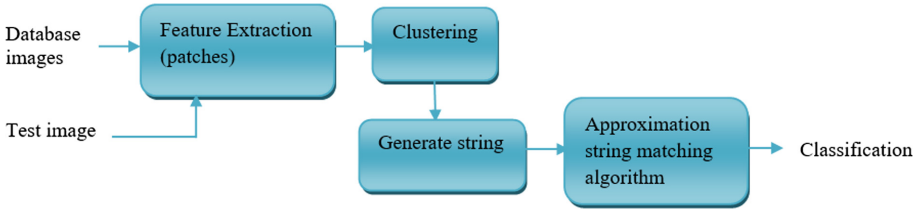

Fig. 1. The architecture of the proposed face recognition system

In the clustering phase, a clustering algorithm is applied on patch vectors to group related patch vectors to the same cluster. Clustering associates each patch vector with a cluster label. There are several clustering algorithms for clustering the data. In this work, K-means clustering is used. In the generation of string phase, a string fora database image is formed by the sequence of cluster labels assigned to patch vectors of a database image. A test mage is also split into a smaller number of patches. Test and database image patch vectors are compared by finding the local distance between them. Cluster label of the database image patch vector\with minimum local distance is obtained and cluster labels with a minimum local distance taken in sequence produce a string for a test image. String generation algorithm is presented in the previous work [34]. Figure 2 shows a sample face image and its corresponding string which is a sequence of cluster labels assigned to patches of the image, the length of the string is 64 which is equal to the number of patches in that image.

In the matching phase, the approximated string matching algorithm is applied to compute the distance between the test and the database image strings and to classify the test image based on the minimum distance computed. The algorithm performs classification in addition to the matching of face images. Thus, the approximation algorithm using distance measures for face image matching does not require any classifier. Hence the complexity of the system is less.

"2 69 12 46 20 46 70 8 17 18 46 20 20 20 43 15 17 61 54
44 28 43 61 55 10 38 70 68 19 22 11 15 30 44 46 20 39 28
22 63 3 62 19 13 60 28 12 63 17 3 62 22 11 35 12 48 17
15 3 27 62 22 63 64"

Fig. 2. Example of a face image and its corresponding string

3.2 Approximate String Matching Algorithm Based on Dynamic Edit Distance

This is also known as Levenshtein distance, which counts the number of dissimilarities between two strings [15]. The Edit distance between two stringss$_1$ and s$_2$ is Ed (s$_1$, s$_2$), defined as the least number of operations required to transform s$_1$ into s$_2$. The operations are transformation (substitution), insertion, and deletion.

Two strings s$_1$ = "appreciate" and s$_2$ = "approximate", the edit distance between s$_1$ and s$_2$ is 3 since it requires 2 substitutions and 1 insertion to transform s$_1$ into s$_2$. This is shown in Fig. 3.

Fig. 3. String matching based on edit distance

Edit distance for image matching is computed using the following recurrence relation Eq. 1.

$$ED(i,j) = \begin{cases} i;\ if\ j = 0 \\ j;\ if\ i = 0 \\ ED[i-1, j-1],\ if\ i, j = 0\ and\ x_i = y_j \\ \min(ED[i-1, j-1] + 1,\ ED[i-1, j] + 1,\ ED[i, j-1] + 1), \\ if\ i! = j\ and\ x_i \neq y_j \end{cases} \quad (1)$$

Algorithm 1. Dynamic EDIT distance for image matching

Input: TestT and Database D image strings of lengths n, m respectively
Output:
Edit_dist: the edit distance between test and database image;
Set each element in ED to 0
ED [i, 0] = i; ED [0, j] = j;
fori = 1 ton do
forj = 1 to m do
if (T(i-1) = = D(j-1))
ED (i,j)= ED(i-1,j-1);
else
ED(i,j) =min{ED(i-1,j-1)+1,ED(i-1,j)+1,ED(i,j-1)+1};
end if
end for
end for
Edit_dist= ED (n,m);
returnEdit_dist;

In Algorithm 1, T and D are test and database face image strings. String element of T is compared with the string element of D if matches EDIT distance is set to 0, otherwise, it is set to a minimum of insert, delete, and substitute operation costs. The time complexity of the EDIT distance algorithm is $O(mn)$, where m is a length of database image string and n is the length of the test image string.

3.3 Approximate String Matching Algorithm Based on Dynamic LCSS Distance

It is a measure of similarity between two strings, which is determined by finding common substrings between them [16]. The pattern is said to match a certain string in a string sequence if the pattern has the maximum number of similarities with that string compared to other strings. In the Longest common subsequence algorithm, the goal is to produce their longest common subsequence: the longest sequence of characters that appear left-to-right order in both the strings.

Two string sequences $s_1 = a_1a_2—a_p$ and $s_2 = b_1b_2—b_q$ of lengths p and q respectively, The Longest common subsequence problem computes the length of the largest string that is common to two sequences s_1 and s_2. LCSS for string matching is implemented through dynamic programming. For example, X = PQPRQNQP and Y = QRRMNQPR. The LCSS of X and Y is QRNQP with length 5. Figure 4 describes this example.

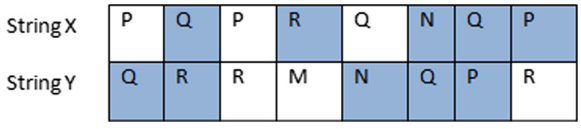

Fig. 4. String matching based on LCSS distance

LCSS distance for image matching is computed using the following recurrence relation Eq. 2.

$$LC(i,j) = \begin{cases} 0, \; if \; i = 0 \, or \, j = 0 \\ LC\left[i-1, j-1\right] + 1, \; if \; i, j > 0 \; and \; x_i = y_j \\ \max\left(LC\left[i,j\right], LC\left[i,j-1\right], LC\left[i-1,j\right]\right), \; if \; i, j > 0 \; and \; x_i \neq y_j \end{cases} \tag{2}$$

Algorithm 2. Dynamic LCSS for image matching

Input:TestT and Database D image strings of lengths n, m respectively
Output:

```
LCSS_dist: the longest common sub sequence distance between test and database images;
Set each element in LC to 0
LC [:,0] = 0; LC[0,:] = 0;
fori = 1 to n do
for j = 1 to m do
if (T(i-1) = = D(j-1))
LC (i,j) =LC (i-1,j-1)+1;
else
      LC(i,j)= max {LC(i-1,j-1),LC(i-1,j),LC(i,j-1)};
end if
end for
end for
LCSS_dist = LC (n,m);
returnLCSS_dist;
```

In Algorithm 2, the String element of T is compared with the string element of D, if they are equal LCSS distance is set to diagonal element cost plus 1, otherwise, it is set to a maximum of up, down, and diagonal element costs. If *m, n* are the lengths of database and test image strings, then the complexity of the LCSS algorithm is $O(mn)$.

3.4 Approximate String Matching Algorithm Based on Hamming Distance

For two similar-length strings X and Y, Hamming distance is the number of character positions at which the corresponding characters in strings are different [17]. Hamming distance determines the least number of replacements or the least number of errors required for transforming a string into another string. The main applications of the hamming code include coding theory and error detection/correction.

For example, given two input strings, s_1 and s_2 Hamming distance is $Hd(s_1, s_2)$. If s_1 = "mouse" and s_2 = "mouth", then $Hd(s_1, s_2) = 2$. The following algorithm is used to calculate the hamming distance between two strings.

Algorithm 3. Hamming distance for image matching

Input:TestT and Database D image strings of equallength n.
Output:
Hd_dist: the Hamming distance between test and database images;
i=0; replace_char=0; //Intialization
while (T(i) !='\0')
if (T(i) != D(j))
replace_char=replace_char+1;
i=i+1;
end if
endwhile
Hd_dist = replace_char;
returnHd_dist;

In Algorithm 3, if the test image string element (T(i)) does not match with the database image string element (D(j)), then the hamming distance is set to the substitution operation cost plus 1. The final hamming distance is based on the number of substitutions or replacements. The time complexity of the hamming distance is $O(n)$, where n is the length of the database as well as test image strings.

3.5 Approximate String Matching Algorithm Based on Jaro Distance

Jaro similarity measure between two strings is determined based on common characters and character transpositions [18]. The maximum Jaro distance results in more similarity between strings. Jaro distance is a normalized score with value 0 for no match and 1 for the correct match. This distance is computed in two steps; the first is the matching and the second is the transposition.

Matching. In this, a score is computed based on the common character occurrences in the strings. The algorithm finds the common characters in two strings within a specified match range around the character's position in the first string with the consideration of unmatched characters in the second string. If s_1 and s_2 are the two strings, then the match range is computed using the following equation.

$$match\_range = \frac{\max(length(s_1), length(s_2))}{2} - 1$$

Transposing. This evaluates the characters in the first string that do not match with the character at the same position in the second string. The transposition score is obtained by dividing this result by 2.

The Jaro distance $Jaro(i, j)$ between 2 strings s_1 and s_2 is

$$Jaro(i,j) = \begin{cases} 0;\ if\ m\_c\ =\ 0 \\ \frac{1}{2}\left(\frac{m\_c}{n} + \frac{m\_c}{m} + \frac{m\_c - T}{m\_c}\right), \\ otherwise \end{cases}$$

Here m_c is the total number of matches.

T is half of the total transpositions.

n, m are the lengths of two strings.

The strings s_1 and s_2 have two characters that match when they are the same and not beyond the match range which is equal to $max(n, m)/2 - 1$.

Algorithm 4. Jaro distance for image matching

Input: TestT and Database D image strings of lengths n, m respectively
Output:

```
Jaro_dist: the Jaro distance between test and database image;
Matc=0; trans=0; //Intialization
Dist_match=(max(n,m)/2)-1
fori = 0 to n do
    f=max(1,i-Dist_match);
    e=min(i+Dist_match+1,m)
forj = f to e do
if(D_matches(j))  continue;
if (T(i) ! = D(j)) continue;
I_matches(i)=1;
D_matches(j)=1;
matc++;
break;
end if
end for
end for
k1=0;
for k = 0 ton do
if(!I_matches(k))  continue;
while(!D_matches(k1)  k1++;
if (T(k) ! = D(k1)) continue;
I_matches(i)=true;
D_matches(j)=true;
trans++;
k1++;
end if
end for
Jaro_dist = ((matc/n) + (matc/m)+((matc – transposition) /match))/2.0
returnJaro_dist;
```

Jaro-Winkler Distance. This similarity measure between two strings is based on character transpositions to a degree adjusted upwards for common prefixes [19]. The Winkler's modification to Jaro distance acts as a boost on the matching score based on the comparison of the string prefixes. Jaro-Winkler algorithm yields a matching score between 0.0 and 1.0.

4 Experiments and Analysis of Results

The approximation string matching algorithms using distance measures on the face recognition system is evaluated using the face databases such as FEI, and AT&T. Matlab is used for implementing the algorithms. The face databases are stored as mat files. All the experiments were run on a 3.6.5 GHz IntelCore i7 CPU with 16 GB RAM.

The Brazilian face database FEI [35] consists of 2800 face images with varied poses and expressions. A subset of this database [36] has cropped and frontal faces of 360 × 260 pixels size and 200 subjects, each with two frontal face images with two different expressions include neutral and smiling expression. FEI sample images with these expressions are shown in Fig. 5 and Fig. 6. AT&T database [37] has a total of 400 images from 40 subjects (10 images in each subject). This database face images have different pose, lighting, and expression changes. Figure 7 represents sample images from the database.

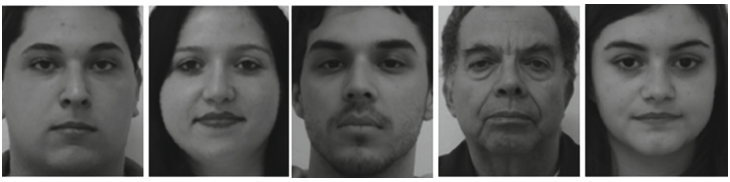

Fig. 5. Face images with neutral expression from the FEI database

Fig. 6. Face images with smile expression from the FEI database

Fig. 7. Sample face images of the AT&T database with different expressions, poses, and lighting

In all the algorithms, a patch size of 8 × 8 pixels is used, the face images are resized to 64 × 64 pixels and the total number of patches in each image 64 and the number of cluster centers (value of K) is chosen to be the nearest value equal to the number of patches in an image. By empirical observation, the number of clusters is fixed to 70 which has given a good performance.

4.1 Face Recognition with FEI Database

The approximation algorithms are evaluated with the FEI database of 400 images of 200 subjects, each subject has one neutral expression face and a smile expression face. A gallery set of 200 neutral face images and a testing set of 200 smiling face images are used for performance evaluation. The images are resized to 64 × 64 and divided into patches of size 8 × 8. The following table describes the comparison between different approximation string matching algorithms and traditional methods Eigen-faces and ICA. The results can conclude that syntactic approaches such as string matching algorithms work well for the face recognition system. All the algorithms except Jaro-Winkler have superior performance over the methods Eigen-faces and ICA. Thus, the approximation string matching algorithms give a better recognition rate than traditional methods for face recognition system (Table 1).

Table 1. Face recognition using the approximation string matching algorithms for the FEI database

Method	No. of gallery images	No. of test images	No. of correct matches	Recognition rate (%)
EDIT	200	200	191	95.5
LCSS	200	200	190	95
Hamming	200	200	192	96
Jaro	200	200	186	93
Jaro-Winkler	200	200	176	88
Eigen-faces	200	200	176	88
ICA	200	200	149	74.5

4.2 Face Recognition with AT&T Database

The performance of the algorithms is evaluated with the AT&T database, which consists of 400 images of 40 subjects. The first five images of all the subjects are used for the gallery database set and the next five images of the subjects are chosen for the testing set. The comparison of the approximation string matching algorithms and the methods Eigen-faces and ICA is described in Table 2. EDIT distance algorithm exhibits a higher recognition rate than other algorithms. Thus, an approximation string matching algorithm using Edit distance performs better for face recognition applications than all other algorithms. Approximation algorithm using hamming distance performs well for recognizing faces with smile expression, but not yields the same performance for the recognition of faces with different poses, lighting, and expression variations. LCSS, hamming, Jaro, and Jaro-Winkler algorithm's performance is average when compared to the EDIT distance algorithm. Face recognition with the AT&T dataset shows that approximation string matching algorithms outperform traditional methods.

Table 2. Comparison between the approximation string matching algorithms and traditional methods for AT&T database

Method	No. of gallery images	No. of test images	No. of correct matches	Recognition rate (%)
EDIT	200	200	187	93.5
LCSS	200	200	176	88
Hamming	200	200	178	89
Jaro	200	200	175	87.5
Jaro-Winkler	200	200	168	84
Eigen-faces	200	200	156	78
ICA	200	200	151	75.5

5 Conclusion and Future Work

In this work, Face recognition is addressed using approximate string matching algorithms using different distance measures such as EDIT, LCSS, hamming, and Jaro. String sequences for the faces are generated and approximated string-based distance is calculated. The string algorithms find the minimum matching path based on the minimum distance and finally, classifies the input face image. The approximation string matching using edit distance yields better results over the other distance metrics such as LCSS, Hamming, Jaro, and Jaro-Winkler for the face recognition system. This work can be used in security, forensic, and other applications. The work presented in this paper concludes that syntactic string approaches perform well for face recognition applications.

References

1. Zhao, W., Chellappa, R., Rosenfeld, A., Phillips, P.: Face recognition: a literature survey. ACM Comput. Surv. **35**, 399–458 (2003)
2. Kirby, M., Sirovich, L.: Application of the Karhunen-Loève procedure for the characterization of the human face. IEEE Trans. Pattern Anal. Mach. Intell. **12**, 103–108 (1990)
3. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Proceedings of the IEEE Conference on CVPR, pp. 586–591 (1991)
4. Yu, H., Yang, J.: A direct LDA algorithm for high-dimensional data with application to face recognition. Pattern Recognit. **34**, 2067–2070 (2001)
5. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fishy faces: recognition using class specific linear projection. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 711–720 (1997)
6. Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: Face recognition by independent component analysis. IEEE Trans. Neural Netw. **13**, 1450–1464 (2002)
7. Miar-Naimi, H., Davari, P.: A new fast and efficient HMM-based face recognition system using a 7-state HMM along with SVD coefficients. Iran. J. Electr. Electron. Eng. **4** (2008)
8. Geng, X., Zhou, Z., Smith-Miles, K.: Individual stable space: an approach to face recognition under uncontrolled conditions. IEEE Trans. Neural Netw. **19**(8), 1354–1368 (2008)
9. Gaston, J., Ming, J., Crookes, D.: Matching larger image areas for unconstrained face identification. IEEE Trans. Cybern. **49**(8), 3191–3202 (2019)

10. Qiangchang, W., Guodong, G.: LS-CNN: characterizing local patches at multiple scales for face recognition. IEEE Trans. Inf. Forensics Secur. **15**, 1640–1652 (2020)
11. Alhendawi, K.M.A., Baharudin, S.: String matching algorithms (SMAs): survey & empirical analysis. J. Comput. Sci. Manag. **2**(5), 2637–2644 (2013)
12. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
13. Ukkonen, E.: Algorithms for approximate string matching. Inf. Control **64**(1–3), 100–118 (1985)
14. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms, 1 edn. Cambridge University Press, Cambridge (2011)
15. Masek, W.J., Paterson, M.: A faster algorithm computing string edit distances. J. Comput. Syst. Sci. **20**(1), 18–31 (1980)
16. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading (1974)
17. Hamming, R.W.: Error detecting and error-correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
18. Jaro, M. A.: Advances in record linkage methodology as applied to the 1985 census of Tampa Florida. J. Am. Stat. Assoc. **84**(406), 414–420 (1989)
19. Winkler, W.E.: Overview of record linkage and current research directions (PDF). Research Report Series, RRS (2006)
20. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
21. Zhou, Z., Wagner, A., Mobahi, H., Wright, J., Ma, Y.: Face recognition with contiguous occlusion using Markov random fields. In: IEEE International Conference Computer Vision (ICCV), pp. 1050–1057, October 2009
22. He, R., Zheng, W.S., Tan, T., Sun, Z.: Half-quadratic-based iterative minimization for robust sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **36**(2), 261–275 (2014)
23. Peng, Y., Li, L., Liu, S., Li, J., Cao, H.: Virtual samples and sparse representation based classification algorithm for face recognition. IET Comput. Vis. **13**(2), 172–177 (2018)
24. Fritz, K., Damiana, L., Serena, M.: A robust group sparse representation variational method with applications to face recognition. IEEE Trans. Image Process. **28**(6), 2785–2798 (2019)
25. Weng, R., Lu, J., Hu, J., Yang, G., Tan, Y.P.: Robust feature set matching for partial face recognition. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 601–608, December 2013
26. Wei, X., Li, C.-T., Lei, Z., Yi, D., Li, S.Z.: Dynamic image-to-class warping for occluded face recognition. IEEE Trans. Inf. Forensics Secur. **9**(12), 2035–2050 (2014)
27. Chen, W., Gao, Y.: Face Recognition Using Ensemble String Matching. IEEE Trans. Image Process. **22**(12), 4798–4808 (2013)
28. Gao, Y., Leung, M.K.H.: Human face profile recognition using attributed string. Pattern Recognit. **35**(2), 353–360 (2002)
29. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2013)
30. Mehdipour Ghazi, M., Kemal Ekenel, H.: A comprehensive analysis of deep learning-based representation for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 34–41 (2016)
31. Lu, J., Wang, G., Zhou, J.: Simultaneous feature and dictionary learning for image set based face recognition. IEEE Trans. Image Process. **26**, 4042–4054 (2017)
32. Hu, G, Peng, X.Y., Hospedales, Y., Verbeek, T.M., Frankenstein, J.: Learning deep face representations using small data. IEEE Trans. Image Process. **27**, 293–303 (2018)
33. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. **29**, 2352–2449 (2017)

34. Krishnaveni, B., Sridhar, S.: Approximation algorithm based on greedy approach for face recognition with partial occlusion. Multimed. Tools Appl. **78**, 27511–27531 (2019)
35. Tenorio, E.Z., Thomaz, C.E.: Analisemultilinear discriminate deformas frontalis de imagens 2D de face. In: Proceedings of the X Simposio Brasileiro de Automacao Inteligente, SBAI, Universidade Federal de Sao Joao del Rei, Sao Joao del Rei, Minas Gerais, Brazil, pp. 266–271, September 2011
36. https://fei.edu.br/~cet/facedatabase.html
37. The Database of Faces, AT&T Laboratories Cambridge (2002). http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html