

# Efficient smart distributed face identification using the MixMaxSim decision function

Sayed Mohammad Ahmadi, Rouhollah Dianat

Department of Computer Engineering and Information Technology, Faculty of Engineering, University of Qom, Qom, Iran

## Article Info

### Article history:

Received Mar 14, 2024

Revised Jul 22, 2024

Accepted Aug 6, 2024

### Keywords:

Clustering  
Deep learning  
Distributed learning  
Face identification  
Facial recognition

## ABSTRACT

Recognizing a large number of people is a common challenge in face identification applications, involving decreased accuracy, increased memory and time complexities. To address these issues, this study introduces a three-module approach: “toilers,” “affinity-meter,” and “decision-maker.” Unlike the random distribution methods used in previous solutions, this method employs clustering to distribute the problem into subnetworks called “toilers.” The toiler’s module calculates the likelihood of test data belonging to each class of each toiler, using the last layer outputs of deep learning models. Meanwhile, the affinity-meter module determines the similarity between the test data and the average of each class, employing a similarity measure. The decision-maker module combines the reports from the previous two modules and selects the final class, utilizing a mix of the max-max criterion and the similarity criterion. The proposed method outperforms existing solutions, achieving improved recall, precision, and F1-score. It effectively addresses memory, speed, and accuracy issues in face identification, surpassing both no-distribution and random methods on Glint360K, VGGFace2, and MS-Celeb-1M datasets. Overall, this method offers a more efficient and accurate approach by distributing the problem into subnetworks, demonstrating superior performance and scalability for large-scale face recognition applications.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Sayed Mohammad Ahmadi

Department of Computer Engineering and Information Technology, Faculty of Engineering, University of Qom  
Qom, Iran

Email: sm.ahmadi@stu.qom.ac.ir

## 1. INTRODUCTION

Facial recognition has been an active research area in machine vision [1], [2], encompassing various subfields such as face detection [3], [4], alignment [5], anti-spoofing [6], [7], and recognition [8], [9]. Face recognition involves both face identification and verification, where the former involves identifying the person in an input image and the latter involves verifying whether two input images belong to the same person. Face recognition has recently gained significant attention as a means of biometric authentication due to the coronavirus disease 2019 (COVID-19) pandemic [10], [11]. Deep learning techniques, especially convolutional neural networks (CNNs), have been at the forefront of face recognition methods for over a decade [12].

Many applications of machine vision and face processing share common challenges with face recognition, including varying light conditions such as brightness and contrast, different poses, or various appearances of the same object in multiple images. Additionally, face recognition is plagued by specific challenges, such as differences in facial expressions, makeup, and age in various images of a single person. Moreover, there are times when intra-class distances exceed inter-class distances [13]. Unlike objects in

object detection that have a clear distinction from each other, faces in face recognition have a very similar structure to each other. The more individuals (classes) involved in face recognition, the more general problems arise due to these common challenges and specific facial features: i) memory consumption problem, ii) increased temporal complexity, and iii) increased accuracy drops

For a large-scale face identification application, the weight matrix requires more memory than most current graphics processing units (GPUs) can handle because the classifier layer in the model is fully connected to the previous layer. Many parameters in this part alone impose a heavy computational burden on the network to compute the loss function and update the weights. Additionally, the large number of neurons in the output layer results in significant overfitting on the network. Due to the specific nature of the face structure, the more classes there are, the more pronounced the drop in accuracy. When all data are trained and identified together, there is a high degree of data dispersion, making it more challenging to distinguish difficult, *i.e.*, similar data. Adopting a distributed approach automatically solves the problems of memory and time. This study aims to propose a distributed approach that does not sacrifice accuracy and, if possible, even improves it.

Two general approaches have been taken to address the challenges posed by face recognition with large-scale datasets. The first approach involves the development of new architectures or loss functions, which have led to the creation of innovative techniques such as L-Softmax [14], A-Softmax [15], NormFace [13], CosFace [16], and ArcFace [17], SFace [18], GhostFace [19]. Although these efforts have yielded promising results, they are often not practical for very high numbers of classes and must be combined with additional techniques. The second approach involves techniques such as softmax dissection [20], active class selection [21], or sampling [22], which have been employed to make it possible to implement the first approach for very high numbers of classes. Although these techniques may result in a slight drop in accuracy, they have proven useful in making it feasible to use new architectures or loss functions for large-scale datasets. The second approach focuses on decomposing the problem into smaller subproblems through divide-and-conquer techniques. Error-correcting output model (ECOC) [23], label mapping [24], independent softmax model (ISM) [25], multi-cognition softmax model (MCSM) [26] are some methods of this approach. The ECOC technique partitions an N-class network into multiple parallel two-class networks. Each binary network can function as a clustering method to categorized data samples. The original class label can be obtained by combining the results of binary class networks. However, this approach requires establishing a correspondence between binary and original network labels. Furthermore, determining the number of binary-class networks and ensuring independence among rows and columns are crucial to minimizing errors. Although this method has not been widely used for face recognition, it could provide valuable insights for related studies. Label mapping is a variant of the ECOC approach that divides the N-class network into larger subnetworks instead of binary-class subnetworks. The subnetworks can have either equal or mixed sizes, and the label mapping approach can be applied accordingly. While this method has not been directly used for face recognition, it could inspire future research in the area.

The ISM method [25], which involves randomly distributing classes into subnetworks, is one such example of the second approach. However, the random distribution of classes can lead to errors, detailed in the proposed method section. one solution to clear this type of error is MCSM. MCSM provides a description of the entire architecture used in [25] as one of its components, termed a cognition unit. It then concurrently trains several cognition units. All cognition units receive test data. Using a voting mechanism, the predicted class is determined by the class with the most votes among multiple cognitive units. MCSM was proposed to address the problem with [25] and mitigate the impact of errors caused by an improper random distribution. MCSM outperforms ISM methods in terms of accuracy. However, it consumes more memory and requires more time for training and prediction.

This study proposes a better solution for the mentioned type of error. It utilizes intelligent distribution of classes to subnetworks through clustering. The clustering algorithm groups similar classes together, leading to a reduction in data dispersion within each cluster. With increased focus on similar data, each subnetwork is expected to improve accuracy. Our proposed method includes three modules: toilers, affinity-meter, and final decision maker. The toiler's module uses supervised neural networks to classify facial images into  $N/m$  classes, where  $N$  is the total number of classes and  $m$  is the number of toilers. The affinity-meter calculates the similarity between the input image and the average features of each class for each toiler. The final decision maker selects the best class from each toiler and determines the final output. Our approach offers several advantages over traditional facial recognition methods. By distributing the workload among multiple toilers, we can significantly reduce the time and memory complexities. Additionally, the use of deep learning allows for a more accurate classification of facial features, leading to improved performance. Finally, our method can easily scale to accommodate larger datasets and more complex classification tasks. Overall, our proposed method offers a powerful and efficient approach to facial recognition. We believe that this method will pave the way for the development of more accurate and reliable facial recognition systems that can be used in a wide range of applications.

The main contributions of this study are: i) Proposing a novel method to enhance face identification accuracy by integrating clustering algorithms with deep learning models; ii) Evaluating the proposed method on three widely used face recognition datasets (Glint360K, VGGFace2, MS-Celeb-1M) and demonstrating significant improvements in accuracy compared to competitive methods; iii) Employing state-of-the-art and robust backbones, such as ArcFace, SFace, and GhostFace, for facial feature extraction; and iv) Conducting a comprehensive experimental analysis of the proposed approach, including comparisons with competitive methods.

**2. PROPOSED METHOD**

The limitation of ISM [25]. This limitation has been generally mentioned in [26] and is thoroughly examined in [27]. Considering that the proposed method is based on the ISM approach [25], we discuss the limitations of the ISM method here. ISM method has a weakness that arises from the random distribution of classes among subnets. To help understand this weakness, consider two scenarios. In the first scenario, we can make the following assumptions:

- a. The problem consists of four classes: x, y, z, and w.
- b. The test sample t belongs to class x.
- c. There are two subnets: A and B.
- d. Classes x and w are situated within subnet A, while classes y and z reside within subnet B.
- e. Classes x, y, and z bear striking resemblances to one another, yet they diverge significantly from class w, as visually depicted in Figure 1.

When we input the test sample into both subnetworks, we anticipate that subnetwork A will yield output values of approximately 0.99 and 0.01, respectively. This expectation arises from the close resemblance between the test sample (t) and class x, while it significantly differs from class w. Conversely, in subnetwork B, we foresee output values around 0.48 and 0.52, as t exhibits similarity to both classes y and z. As a result of the max-max approach, subnet A is chosen from the two subnetworks, and class x is selected, which is the correct choice. This scenario is a *fortunate* status for the random distribution of classes. In the second scenario, we assume that all the assumptions of the first scenario are met, except for assumption d, which changes as follows: dd) x and y belong to subnet A, and w and z belong to subnet B, see Figure 2.

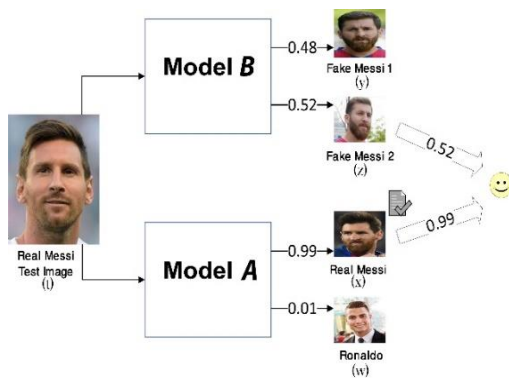


Figure 1. Fortunate in ISM [27]. The test image is predicted true in random distributed scenario

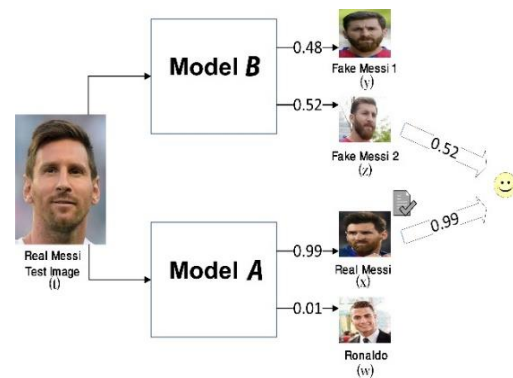


Figure 2. Unfortunate in ISM [27]. The test image is not predicted in random distributed scenario

When the test sample t is fed into both subnets in the second scenario, the output values in subnet A will be approximately 0.52 for class x and 0.48 for class y, while the output values in subnet B will be 0.80 for class w and 0.20 for class z. Therefore, selecting class w within subnet B is an incorrect choice. This phenomenon occurs because the score of the output value of the actual class is randomly reduced since it is placed within a subnet of similar classes. However, similar negative classes within other clusters get higher scores without any serious competitors due to the max-max mechanism, causing the true class to ultimately fail. We call this scenario an *unfortunate* status. To fall into the trap of unfortunate, two conditions must be met together:

*Condition A:* The actual class should be in the same subnetwork as similar classes.

*Condition B:* Similar classes should be distributed in different subnetworks.

One approach to overcoming the weakness of ISM is to use clustering to prevent the occurrence of Condition B. This means that efforts are made to prevent similar classes from being distributed across different clusters and, instead, to place them in a unified cluster. Clustering algorithms, such as k-means, can be used for this purpose. This approach may seem to result in severe competition within clusters and low output values for winning classes. However, this perception is not accurate as clustering actually decreases the dispersion of data within each cluster and allows the network to focus more on distinguishing between similar classes. Thus, the negative effect is neutralized by this positive property.

In contrast to ISM, this method can determine the cluster to which the input image belongs. For each cluster, a representative can be determined and compared with the feature vector of the input image to assign it to the cluster with the highest similarity. This is not possible with ISM, which relies on the max-max mechanism. Therefore, an alternative criterion needs to be used. We use nearest neighbor with similarity measure in the *affinity-meter* module, instead of cluster centers. Figure 3 shows the main idea of this study: Figure 3(a) show pre-processing phase involves distributing all classes into multiple toilers using clustering. Each toiler contains similar classes. In Figure 3(b) test phase, two candidate categories are considered: (b1) predicting the test sample using each subnet (toiler's module) and selecting the class with the highest output value (max-max), and (b2) determining the closest class to the sample based on the affinities of the toiler (nearest class). Finally, (b3) the predicted class is chosen to maximize the product of the output value and the similarity.

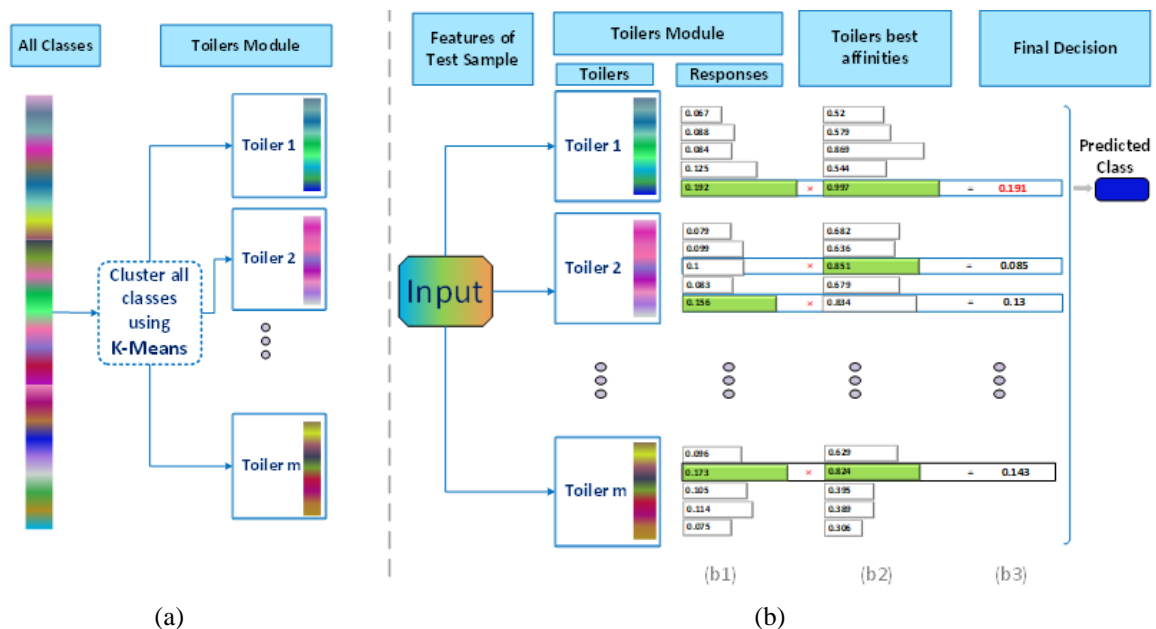


Figure 3. The main concept in the study (a) preprocessing phase, and (b) postprocessing phase

## 2.1. Framework overview

We begin by using a pre-trained CNN [28], [29] to extract features for all images in each dataset, resulting in a collection of features denoted as  $F_D = \{f_i\}_{i=1}^N$ , where  $f_i \in \mathbb{R}^d$ ,  $N$  represents the number of images,  $D$  is a dataset, and  $d$  represents the feature vector dimension. To implement our face identification architecture, we break the identification problem into three modules. The first module comprises a group of toilers, with each being trained to identify a specific subset of classes. Clustering is used to distribute all classes among the toilers, ensuring that each toiler is responsible for identifying similar classes. During testing, the extracted feature vector from the test image is given as input to all toilers. Each toiler (which is a neural network) calculates and returns the output values associated with each class for the test data. The second module, the *affinity-meter* module, calculates the similarity of the input to the average features of each class of each toiler separately. Finally, the third module, the *final decision maker* module, summarizes the reports received from the first and second modules and selects a class as the final category that has the highest overall score obtained from the first module's output value and the affinity score from the second module.

## 2.2. Framework modules

Our proposed method consists of three modules: i) toilers, ii) affinity meter, and iii) final decision maker. Each module plays a crucial role in ensuring accurate and efficient outcomes. This modular architecture allows for specialized processing while maintaining a clear flow of information between components.

### 2.2.1. Toilers

Each toiler is a supervised neural network that works in parallel with other toilers and is responsible for identifying approximately  $N/m$  classes, where  $N$  is the total number of classes and  $m$  is the number of toilers. The distribution of classes among these toilers is performed using clustering algorithms such as k-means, which groups similar classes in the same cluster and dissimilar classes in different clusters. Each toiler is trained using data corresponding to its classes, which are the features of images extracted by a pre-trained *LResNet100-IR* model. During the test phase, the toiler's module determines the similarity of the input data to each of its assigned classes and sends the results to the next module for further processing.

#### a. Time complexity

With a 512-dimensional input feature vector ( $d = 512$ ) and each toiler handling approximately  $N/m$  classes, the weight matrix in the pre-final layer has dimensions of  $d \times N/m$ . Given that each toiler is trained solely on data relevant to its assigned classes and the entire dataset size is  $D$ , each toiler is trained with about  $D/m$  data. This leads to a total time complexity for each epoch (Epoch Time):

$$ET = \frac{D}{m} \times d \times \frac{N}{m} = d \times N \times \frac{D}{m^2} \quad (1)$$

If the toilers are trained in parallel on multiple machines, the *Total Time required in Parallel Mode* can be calculated as (2):

$$TTPM = e \times \frac{D}{m} \times d \times \frac{N}{m} = d \times e \times N \times \frac{D}{m^2} \quad (2)$$

However, if they are trained sequentially, the *Total Time required in Serial Mode* need to be calculated as (3):

$$TTSM = e \times d \times \frac{N}{m} \times D \quad (3)$$

If only a single toiler is employed ( $m=1$ ), the necessary number of time units will be:

$$TTOM_{(m=1)} = TTSM_{(m=1)} = e \times d \times N \times D, \quad (4)$$

which is  $m$  times slower than the distributed model in serial running and  $m^2$  times slower than the distributed model in parallel running. Hence, the greater the number of toilers, the fewer total operations are needed. But on the other hand, the risk of overfitting will increase. Therefore, a balance point should be considered between the complexity of time (the number of required operations) and the accuracy of the system.

#### b. Memory complexity

Each toiler needs to be trained for  $e$  epochs, with each step of each epoch requiring  $b$  batches of data. These batches are multiplied by the weight matrix with dimensions  $512 \times C/m$ , resulting in a  $C/m$  vector. Therefore, in each step, we require  $b \times 512 \times 4$  bytes to store input data and  $512 \times C/m \times 4$  bytes for storing weight matrix.

### 2.2.2. Affinity meter

This module calculates the similarities between the input data and the average features of each class for each toiler. First, we calculate the average feature vectors of training data for each class:

$$\bar{F}_{c_i} = \frac{\sum_{j=1}^{n_i} f_j}{n_i}, i = 1, 2, 3, \dots, C, \quad (5)$$

where  $c_i$  denotes the specific class in the dataset,  $\bar{F}_{c_i}$  is the mean of feature vectors of class  $c_i$ ,  $C$  is the total number of classes in the dataset,  $f_j$  is the  $j^{\text{th}}$  feature vector of the class  $c_i$  from the training data,  $n_i$  is the number of training samples for class  $c_i$ . Next, we calculate the affinity matrix based on the similarity between the input feature vector and the mean feature vectors of all classes for each toiler.

$$A_{in}^t = \cos(f_{in}, \bar{F}_{c_i}^t),_{(t=1,2,\dots,m)},_{(i=1,2,\dots,C_t)}, \quad (6)$$

where  $f_{in}$  represents the features of the input sample,  $m$  denotes the number of toilers,  $C_t$  is the number of classes in  $t$ -th toiler.  $A_{in}^t$  represents the affinity between the input and all classes in toiler  $t$ , and  $\bar{F}_{c_i}^t$  denotes the average feature vector of class  $c_i$  for toiler  $t$ .

a. Time complexity

If we assume that we have  $n$  images for each identity, calculating the mean of these  $n$  feature vectors will require  $n \times 512$  operations. Additionally, calculating the affinity between any two feature vectors requires 512 operations. With  $C$  classes, the total number of operations needed is  $512^2 \times n \times C$  operations.

### 2.3.3. Final decision maker

This module is responsible for selecting the final class label for the input image based on the results obtained from each *toiler*. The following steps are performed for every toiler:

- The module finds the class with the highest output score from the toiler's module. Subsequently, it retrieves the affinity value of that class from the affinity-meter module and multiplies the two values.
- The module also finds the class with the highest affinity value from the affinity-meter module and retrieves its toiler response value from the toiler's module. It multiplies these two values as well.
- The module selects the maximum multiplied value obtained from steps a and b, and the class corresponding to that value is considered as the chosen class of that toiler.

Although the three modules are logically separate, in practice, the first module performs its task and sends the results to the third module. By examining the results of the first module, the third module may decide not to use the second module to speed up performance. This is because if the selected class according to the results of the first module has a very high score, its decision is accurate and true and almost always verified by the second module. The second module is actually used as an auxiliary tool for cases where the first module is uncertain and has a low score. Further details regarding this point will be provided in the experiments section. Algorithm 1 provides a more detailed description of the post-processing step (test phase).

#### Algorithm 1. Post-processing phase

INPUTS: face:  $112 \times 112 \times 3$  RGB face image, c: number of subnets, thr: threshold

OUTPUT:  $y_{pred}$ : predicted class label

```

# Feature extraction
1 F = extract_features(face)

# Best classes from toilers module
2 for m in range(0, c):
3     A_value = max(Model(m, F))
4     A_class = argmax(Model(m, F))
5     A_bestValue = 0.
6     if A_value > A_bestValue:
7         A_bestValue = A_value
8         A_bestClass = A_class
9         A_bestCluster = m
10 C_value = cos_sim(A_bestClass, F)
11 if A_bestValue > thr:
12     return y_pred

# Best classes from affinity-meter module
13 for m in range(0, c):
14     B_class = argmin(cosSim(clustData, F))
15     B_value = cos_sim(B_class, F)
16     if B_value > B_best:
17         B_bestValue = B_value
18         B_bestClass = B_class
19         B_bestCluster = m
20 D_value = Model(B_bestCluster, F)[B_bestClass]

# Final decision
21 If A_bestValue × C_value > B_bestValue × D_value:
22     y_pred = label of A_bestClass.
23 otherwise:
24     y_pred = label of B_bestClass.
25 return y_pred

```

### 3. RESULTS AND DISCUSSION

#### 3.1. Experimental settings

##### a. Datasets

This study used the following datasets: i) The Glint360k database [22] is renowned as the cleanest and most extensive resource for face recognition. Within this database, a remarkable 360,232 unique individuals are represented, accompanied by an impressive collection of 17 million images; ii) MS-Celeb-1M [30] contains 100K identities, each identity has about 100 facial images; and iii) VGGFace2 [31] contains more than 3.3M images of over 9K identities.

For each dataset, the classes were selected randomly. They were divided into three parts: 60% for training, 20% for validation, and the rest for testing. In consideration of the varying number of samples in each class, we, therefore, consider the evaluation sample size equivalent to the number of test samples, i.e., five for each. In the absence of sufficient samples in some classes, new data were generated using data augmentation, through random rotation, flip, brightness, and contrast transformations.

##### b. Feature extraction

The proposed algorithm employs some of the latest and most powerful feature extractors instead of the raw images:

- ArcFace [17]: The backbone architecture of this extractor is LResNet100E-IR [17], which is a residual network [28]. It was trained using the ArcFace loss function and the MS-Celeb-1M [30] dataset.
- Sface [18]: This study employs the IResNet50 version of the feature extractor as the backbone and utilizes CosFace [16] as the loss function. The authors trained this model from scratch using the VGGFace2 [31] dataset.
- GhostFaceNet [19]: This backbone incorporating GhostModules. The utilized version in this study is the first and configured with a width of 1 and a stride of 3.

All backbones take 112×112 RGB images and convert them into 512-dimensional feature vectors. We utilized the pre-trained weights provided in.

##### c. Selection of toilers' representatives

There are some candidates for choosing a representative for each cluster: i) The mean can potentially serve as the representative for each cluster. However, in practice, it may not accurately represent large-scale face features due to the nature of data distribution, refer to Table 1; ii) Alternatively, a strong candidate for being the cluster representative is the closest data point to the sample features. In other words, the test sample belongs to the cluster containing the closest data point, refer to Table 1; and iii) The maximum value of all toiler outputs can be used to determine the cluster to which each data belongs (ISM method).

Table 1. Clustering accuracy

Representative	Train Accuracy %	Test Accuracy %
Mean	67.08	59.72
Nearest	98.08	98.17

\* For 20,000 classes, using k-means with 10 clusters

##### d. Uncertainty area

For a problem involving 20,000 classes, within the Glint360k dataset, most of the test samples yield output values ranging from 0.6 and 0.8. Approximately 80% of the output values of these test samples are exceed a specific threshold (approximately 0.4 for 20,000 classes) in the max-max approach, see Figure 4. Nearly 100% of these values represent correct predictions as shown in Figure 5. In other words, almost all of the network's predictions have been accurate for output values greater than 0.4. When the network's confidence level drops to around 0.4, the error prediction rate increases. Consequently, any proposed solution should primarily concentrate on this particular range. It should be noted that this threshold value may vary depending on the type of dataset, backbone type, loss function, and architectural features. For this reason, we employed a dynamic approach to determine it. In this manner, for validation datasets, we consider the largest value as the threshold, ensuring that at least 99.5% of the data with output values higher than this threshold are correctly predicted. In other words, we find a point where the majority of error-prone data have output values lower than it, and we refer to this region as the uncertainty region. Our method aims to concentrate its focus on this region. For example, in Figure 6, which illustrates the distribution of toiler unit candidates, the threshold value is 0.49, and almost all output values above this threshold have been accurately predicted. Figure 7 depicts the distribution of affinity-meter unit candidate values, which is very similar in shape to Figure 6. This implies that in the uncertainty region, with the collaboration of the opinions of both units, a higher accuracy can be achieved. In fact, whenever the output value of a test data input from toiler unit

exceeds the threshold, the framework no longer awaits the opinion of the affinity-meter unit and relies solely on the toiler unit's report for decision-making. In addition, when considering the similarity between the features of those classes and the test data, it is significantly low. Nevertheless, mixing these two criteria, i.e., similarity and the max-max criterion, can lead to more robust results.

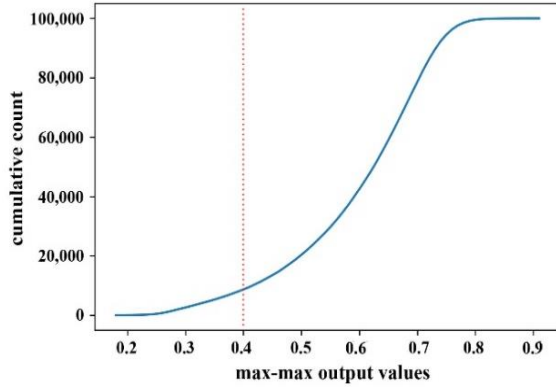


Figure 4. Cumulative frequency of test samples based on the output value, with errors occurring before the red line (uncertainty area)

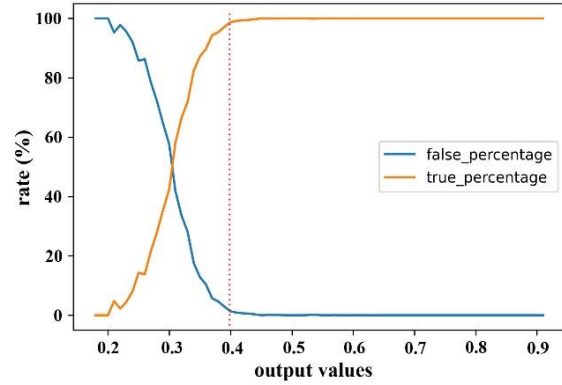


Figure 5. Ratio of correct and incorrect predictions for max-max output values in the test dataset, with incorrect predictions occurring before the red line (uncertainty area)

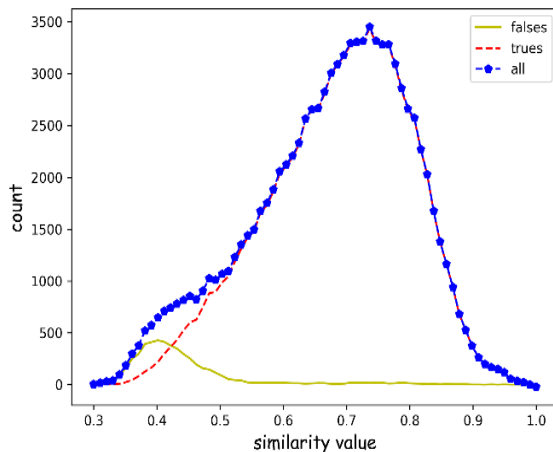


Figure 6. Frequency of each max-max value of toiler's candidates for validation data (GhostFace, MS-Celeb-1M, 20,000 classes, 6 subnets)

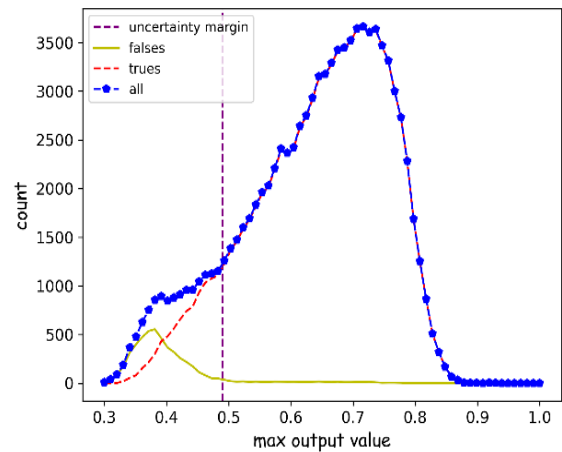


Figure 7. Frequency of each similarity value of affinity-meter candidates for validation data (GhostFace, MS-Celeb-1M, 20,000 classes, 6 subnets)

#### e. Clustering of classes

This study demonstrates that the use of clustering can alleviate the errors caused by the random distribution of data in toiler's module. The more accurate the clustering is performed, the higher the final performance will be. For simplicity and to prove the correctness of our proposed method, we use a very simple clustering algorithm, k-means. According to this approach, clustering was applied on all feature vectors of the training data. However, the problem is that not all samples related to each class are grouped into a single cluster. Nevertheless, to address this issue, each class is assigned to a cluster that has the highest number of samples in it. In the test phase, we can estimate the toiler (cluster) of a test sample using the affinity-meter module. This is because the nearest neighbors of each toiler provide a better measure for estimating the cluster compared to the centers of the clusters. From the mentioned descriptions, it can be inferred that the provided framework is independent of the dataset and also the backbone. Our experiment



results in Table 2 also confirm this claim. These experiments were conducted on the datasets MS-Celeb-1M [30], VGGFace2 [31], with various new backbones including SFace [18], GhostFaceNet [19]. In all of these cases, an improvement in accuracy is observed in the proposed scenario compared to competing methods.

Table 2. Comparison results on GhostFaceNet and SFace

Dataset	Backbone	Classes	Distribution-Free			Toilers	ISM [25]			our pre+ISM post			our pre+our post		
			Pre*	Rec	F1		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
MS-Celeb-1M [30]	GhostFaceNet [19]	5,000	97.19	96.68	96.59	2	97.21	96.74	96.64	97.14	96.66	96.56	97.27	96.80	96.71
		10,000	95.29	94.46	94.27	3	95.93	95.22	95.08	95.85	95.10	94.97	96.00	95.29	95.17
		20,000	94.25	93.16	92.92	6	95.79	95.05	94.91	95.64	94.88	94.73	95.87	95.13	95.00
		50,000	91.19	89.21	88.70	16	95.15	94.23	94.08	94.87	93.90	93.73	95.23	94.32	94.18
VGG-Face2 [31]	SFace [18]	5,000	91.31	90.03	89.73	2	91.60	90.35	90.07	91.47	90.12	89.84	91.99	90.68	90.46
		6,000	90.91	89.50	89.20	3	91.33	89.99	89.73	91.02	89.58	89.31	91.73	90.37	90.15
		7,000	90.51	89.03	88.70	3	90.94	89.55	89.26	90.80	89.34	89.04	91.47	90.07	89.84
		8,000	90.12	88.54	88.23	3	90.72	89.27	88.97	90.60	89.08	88.77	91.21	89.78	89.53
		8,900	89.65	88.05	87.69	4	90.42	88.95	88.62	90.10	88.49	88.16	90.87	89.44	89.17

\* Evaluation metrics are Precision (Pre), Recall (Rec), and F1-score

#### f. Implementation details

A single architecture was employed in all the experiments, featuring a 512-dimensional vector in the input layer, followed by L2 normalization. The final layer, referred to as the classifier layer, consists of  $N_i$  neurons, which  $i$  represents the subnetwork's index. We utilized the ArcFace loss function for training, employing the Adam optimizer with a learning rate of 0.001, a beta1 of 0.9, and a beta2 of 0.99. All scenarios were trained for 50 epochs, and the experiments were conducted with varying class sizes.

#### g. Similarity value or subnetworks output value?

It has been noted that the decision-making unit multiplies the values entered from the previous units for each of these two candidates and selects the class with the higher product. However, experiments indicate that the importance of candidates from toiler and affinity-meter units may not be equal, depending on the type of datasets, training methodologies, loss functions used in the training of subnetworks, and other factors. As the extracted features become more accurate, the affinity-meter unit candidate gains higher importance in decision-making. Therefore, an additional coefficient term, denoted as  $w$ , is introduced to the formula. In other words, assuming: i) The affinity-meter unit candidate class is denoted as  $x$  and ii) The toiler unit candidate class is denoted as  $y$ ,

The selected class will be the one that maximizes the expression (7) among all subnetworks:

$$\max(a \times b \times w, c \times d), \quad (7)$$

where  $a$  represents the output value of  $x$ ,  $b$  represents the similarity value of  $x$ ,  $c$  represents the output value  $y$ , and  $d$  represents the similarity value of  $y$ . As mentioned,  $w$  represents the importance of the affinity-meter unit candidate, ranging between zero and two. If its value is less than one, it signifies a higher importance given to the affinity-meter unit candidate. If it is greater than one, it indicates a higher importance given to the toiler unit candidate.

The question arises: How to determine the value of  $w$ ? We consider 200 potential values for  $w$  in the range of zero to two with intervals of 0.01. Subsequently, we calculate the accuracy for each of these  $w$  values on the validation dataset. The point where the highest accuracy is achieved is then regarded as the final value for  $w$ . As depicted in Figure 8, for the MS-Celeb-1M datasets with 20,000 categories and 6 subnetworks, and for validation datasets, the optimal  $w$  is 1.4, yielding the highest accuracy of 95.131. This same  $w$  is applied to test dataset in Figure 9, and it is observed that nearly optimal results are achieved. The best  $w$  for test datasets is 1.36 (with an accuracy of 95.163), very close to our chosen value of  $w=1.4$  (accuracy of 95.156). Additionally, the figure illustrates that solely utilizing the toiler unit criterion (*i.e.*,  $w=0$ ) would result in an accuracy of 94.837, while using only the affinity-meter unit criterion (*i.e.*,  $w=2$ ) would yield an accuracy of 95.114. The combination of both criteria surpasses the performance of each criterion individually.

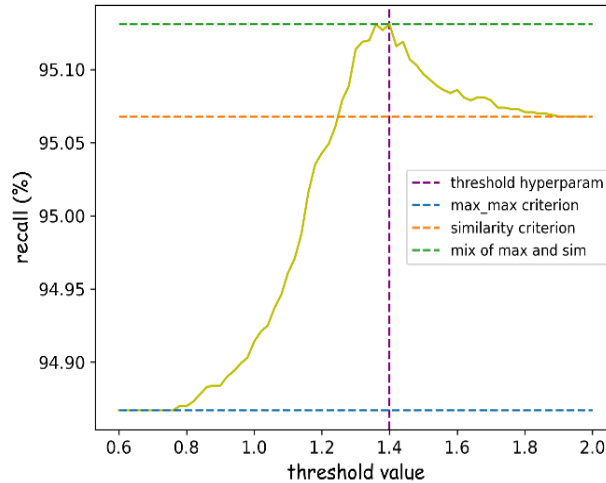


Figure 8. Finding best threshold point from validation dataset with 20000 classes and 6 subnetworks (GhostFaceNet+MS-Celeb-1M)

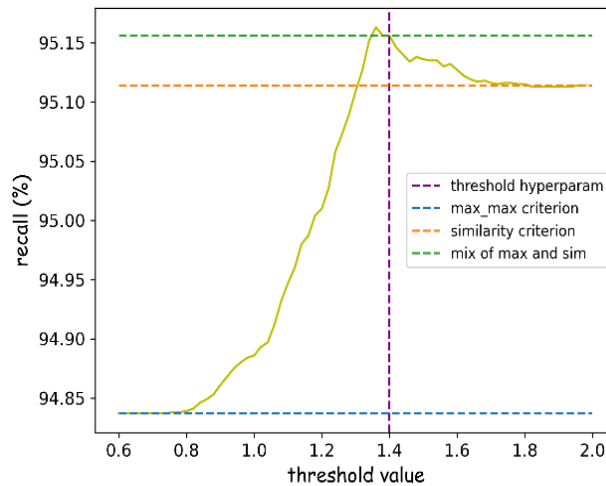


Figure 9. Accuracy (recall) in different potential threshold values and selected threshold value for test dataset (GhostFaceNet+MS-Celeb-1M)

### 3.2. Method comparison

The following face identification scenarios were evaluated in each case:

- Distribution-Free: A single network.
- ISM: Multiple subnets in parallel, random class distribution between subnets, and the max-max prediction mechanism.
- Our *Pre + max – max* Post: Multiple subnets in parallel, distribution classed by clustering, max-max decision method.
- Our *Pre + Our* Post: Multiple subnets in parallel, clustering-based class distribution, and a modified max-max prediction mechanism (proposed method).

Table 3 summarizes the results of face identification for all scenarios. Different results for different datasets can be caused by differences in the level of cleanliness or difficulties of the dataset images. Another reason is that the pre-trained model was trained using MS-Celeb-1M dataset, which is why it has better results than other datasets. Of course, in this case, the proposed method worked better than other methods. When the number of classes is small, there is a lower risk of overfitting, and consequently, the proposed model and other distribution methods may not necessarily lead to performance improvement. However, as the number of classes increases, the proposed method consistently outperforms other methods. According to

Figures 10 to 12. Recall drop for distributive scenarios (Sface [18], VGGFace2 [31]) the accuracy rate decreases with increasing class size in all scenarios. However, the proposed method has a lower accuracy drop. The source code of experiments, is available in [32].

Table 3. Comparison results on ArcFace for all scenarios

Dataset	Classes	Distribution-Free			Toilers	ISM [25]			our pre + ISM post			our pre + our post		
		Pre*	Rec	F1		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Glint-360K [22]	5000	95.58	94.76	94.65	2	95.86	95.09	94.99	95.88	95.09	94.99	95.99	95.26	95.19
	10,000	93.31	91.95	91.74	3	94.49	93.46	93.29	94.53	93.49	93.34	94.79	93.84	93.74
	20,000	89.38	86.54	86.04	6	93.23	91.94	91.74	93.39	92.15	91.96	93.57	92.43	92.28
	50,000	73.21	63.38	61.85	16	90.64	88.87	88.60	90.84	89.19	88.91	91.21	89.68	89.49
MS-Celeb-1M [30]	5,000	93.46	92.26	92.00	2	94.05	92.94	92.73	94.22	93.16	92.94	94.16	93.16	92.94
	10,000	90.42	88.54	88.10	3	91.91	90.46	90.15	91.95	90.60	90.29	91.99	90.63	90.34
	20,000	87.97	85.10	84.44	6	91.56	90.13	89.79	91.81	90.38	90.03	91.80	90.51	90.18
	50,000	74.31	65.30	63.86	16	90.22	88.58	88.16	90.43	88.80	88.40	90.47	88.99	88.58
VGG-Face2 [31]	5,000	89.08	87.18	86.87	2	89.35	87.54	87.24	89.33	87.49	87.18	89.38	87.66	87.38
	6,000	88.50	86.55	86.26	3	89.03	87.23	86.93	89.07	87.31	86.99	89.07	87.31	86.99
	7,000	87.68	85.64	85.28	3	88.52	86.67	86.32	88.65	86.72	86.42	88.61	86.77	86.49
	8,000	87.24	85.05	84.67	3	88.15	86.18	85.82	88.15	86.22	85.87	88.17	86.28	85.96
	8,900	86.54	84.24	83.85	4	87.55	85.63	85.24	87.68	85.76	85.39	87.87	85.93	85.61

\* Evaluation metrics are Precision (Pre), Recall (Rec), and F1-score

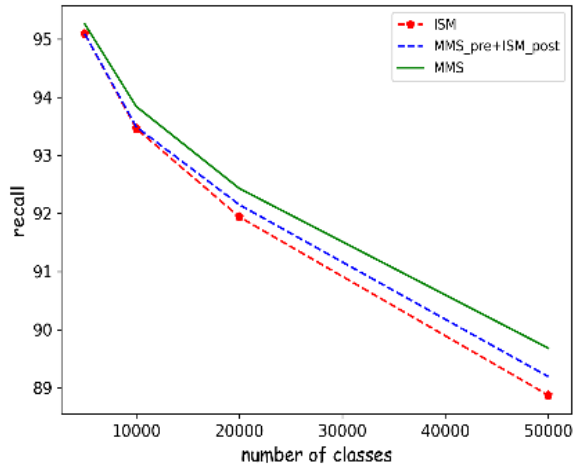


Figure 10. Recall drop for distributive scenarios (ArcFace [17], Glint360k [22])

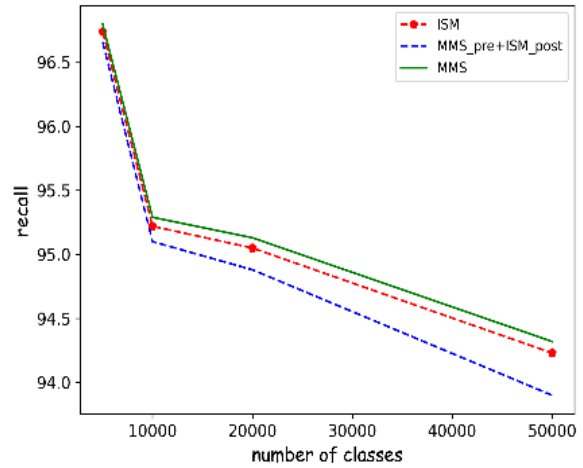


Figure 11. Recall drop for distributive scenarios (GhostFace [19], MS-Celeb-1M [30])

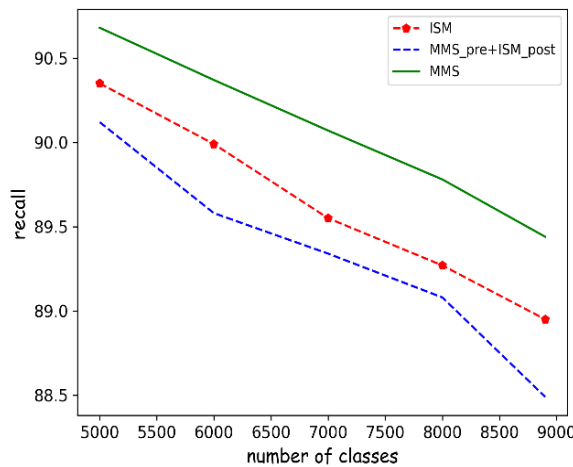


Figure 12. Recall drop for distributive scenarios (Sface [18], VGGFace2 [31])

#### 4. CONCLUSION

In summary, this study comprised two distinct phases: preprocessing and post-processing, both of which introduced no additional complexity to the training operation. To address challenges related to accuracy, speed, and memory, a divide-and-conquer approach was employed, effectively breaking down the problem into smaller subproblems. The classifier layer plays a critical role, especially when dealing with numerous classes within a single network. The risk of overfitting can be mitigated by decomposing a large-scale problem into smaller subproblems. We divided the large problem into subproblems using clustering in the preprocessing phase and combined similarity criteria with subnetwork output values in the postprocessing phase. One promising avenue for future research is the reduction of clustering errors to maximize overall classification accuracy for numerous classes. Additionally, the potential exists to enhance feature extraction robustness by unfreezing the last few layers during retraining of the base model.




#### REFERENCES

- [1] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823, doi: 10.1109/CVPR.2015.7298682.
- [2] M. A. Al Noman *et al.*, "A computer vision-based lane detection technique using gradient threshold and hue-lightness-saturation value for an autonomous vehicle," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 347–357, Feb. 2023, doi: 10.11591/ijece.v13i1.pp347-357.
- [3] T. Tsai and P. Chi, "A single-stage face detection and face recognition deep neural network based on feature pyramid and triplet loss," *IET Image Processing*, vol. 16, no. 8, pp. 2148–2156, Mar. 2022, doi: 10.1049/ipr2.12479.
- [4] T. H. Obaida, A. S. Jamil, and N. F. Hassan, "Real-time face detection in digital video-based on Viola-Jones supported by convolutional neural networks," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 3, pp. 3083–3091, Jun. 2022, doi: 10.11591/ijece.v12i3.pp3083-3091.
- [5] M. Jabberi, A. Wali, B. B. Chaudhuri, and A. M. Alimi, "68 landmarks are efficient for 3D face alignment: what about more?: 3D face alignment method applied to face recognition," *Multimedia Tools and Applications*, vol. 82, no. 27, pp. 41435–41469, Apr. 2023, doi: 10.1007/s11042-023-14770-x.
- [6] R. Huang and X. Wang, "Face anti-spoofing using feature distilling and global attention learning," *Pattern Recognition*, vol. 135, Mar. 2023, doi: 10.1016/j.patcog.2022.109147.
- [7] M. Khammari, "Robust face anti-spoofing using CNN with LBP and WLD," *IET Image Processing*, vol. 13, no. 11, pp. 1880–1884, Jul. 2019, doi: 10.1049/iet-ipr.2018.5560.
- [8] G. Gao, Y. Yu, J. Yang, G.-J. Qi, and M. Yang, "Hierarchical deep CNN feature set-based representation learning for robust cross-resolution face recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2550–2560, May 2022, doi: 10.1109/tcsvt.2020.3042178.
- [9] Y. El Madmoune, I. El Ouariachi, K. Zenkour, and A. Zahi, "Robust face recognition using convolutional neural networks combined with Krawtchouk moments," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 4, pp. 4052–4067, Aug. 2023, doi: 10.11591/ijece.v13i4.pp4052-4067.
- [10] M. I. P. Nasution, N. Nurbaiti, N. Nurlaila, T. I. F. Rahma, and K. Kamilah, "Face recognition login authentication for digital payment solution at COVID-19 pandemic," *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, Yogyakarta, Indonesia, 2020, pp. 48–51, doi: 10.1109/ic2ie50715.2020.9274654.
- [11] J. S. Talahua, J. Buele, P. Calvopiña, and J. Varela-Aldás, "Facial recognition system for people with and without face mask in times of the COVID-19 pandemic," *Sustainability*, vol. 13, no. 12, p. 6900, Jun. 2021, doi: 10.3390/su13126900.
- [12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.
- [13] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L2 hypersphere embedding for face verification," in *Proceedings of the 25th ACM International Conference on Multimedia*, Oct. 2017, pp. 1041–1049, doi: 10.1145/3123266.3123359.
- [14] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," *arXiv:1612.02295*, 2016.
- [15] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: deep hypersphere embedding for face recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6738–6746, doi: 10.1109/CVPR.2017.713.
- [16] H. Wang *et al.*, "CosFace: large margin cosine loss for deep face recognition," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 5265–5274, doi: 10.1109/cvpr.2018.00552.
- [17] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: additive angular margin loss for deep face recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 4685–4694, doi: 10.1109/cvpr.2019.00482.
- [18] F. Boutros, M. Huber, P. Siebke, T. Rieber, and N. Damer, "SFace: Privacy-friendly and accurate face recognition using synthetic data," *2022 IEEE International Joint Conference on Biometrics (IJCB)*, Abu Dhabi, United Arab Emirates, 2022, pp. 1–11, doi: 10.1109/ijcb54206.2022.10007961.
- [19] M. Alansari, O. A. Hay, S. Javed, A. Shoufan, Y. Zweiri, and N. Werghi, "GhostFaceNets: lightweight face recognition model from cheap operations," *IEEE Access*, vol. 11, pp. 35429–35446, 2023, doi: 10.1109/access.2023.3266068.
- [20] L. He, Z. Wang, Y. Li, and S. Wang, "Softmax dissection: Towards understanding intra- and inter-class objective for embedding learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 10957–10964, Apr. 2020, doi: 10.1609/aaai.v34i07.6729.
- [21] X. Zhang, L. Yang, J. Yan, and D. Lin, "Accelerated training for massive classification via dynamic class selection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.12337.
- [22] X. An *et al.*, "Partial FC: training 10 million identities on a single machine," *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Montreal, BC, Canada, 2021, pp. 1445–1449, doi: 10.1109/iccvw54120.2021.00166.
- [23] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, Jan. 1995, doi: 10.1613/jair.105.
- [24] Q. Zhang, K.-C. Lee, H. Bao, Y. You, W. Li, and D. Guo, "Large scale classification in deep neural network with label mapping," *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov. 2018, doi: 10.1109/icdmw.2018.00163.




- [25] Y. Wu, J. Li, Y. Kong, and Y. Fu, "Deep convolutional neural network with independent softmax for large scale face recognition," *In Proceedings of the 24th ACM international conference on Multimedia (MM '16)*. Association for Computing Machinery, New York, NY, USA, Oct. 2016, pp. 1063–1067 doi: 10.1145/2964284.2984060.
- [26] Y. Xu *et al.*, "High performance large scale face recognition with multi-cognition softmax and feature retrieval," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Venice, Italy, 2017, pp. 1898-1906, doi: 10.1109/iccvw.2017.224.
- [27] S. M. Ahmadi and R. Dianat, "A two-stage clustering-based distributed framework for large-scale face identification," *Signal and Data Processing*, vol. 21, no. 1, 2024.
- [28] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 6307-6315, doi: 10.1109/cvpr.2017.668.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [30] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "MS-Celeb-1M: a dataset and benchmark for large-scale face recognition," in *Computer Vision*, Springer International Publishing, 2016, pp. 87–102.
- [31] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, May 2018, pp. 67-74 doi: 10.1109/fg.2018.00020.
- [32] S. M. Ahmadi and R. Dianat, "MixMaxSim: mixture of MaxMax and similarity," *GitHub*, <https://github.com/mohammadahmadi1395/MixMaxSim> (accessed Feb. 10, 2024).

## BIOGRAPHIES OF AUTHORS



**Sayed Mohammad Ahmadi**    was born in Qom, Iran in 1986. is a professional in the field of computer science and information technology. He holds a bachelor of computer science degree from the Faculty of Basic Sciences at University of Qom in 2010. He further pursued his studies and obtained a master's degree in information technology engineering from the Faculty of Engineering at University of Qom in 2014. Currently, he is a Ph.D. student in information technology engineering at the same institution. He has a diverse academic and teaching background, having taught at various universities in Afghanistan and Iran. He has also been involved in data analysis and image processing projects. He can be contacted at email: [ahmadi.mohammad2008@gmail.com](mailto:ahmadi.mohammad2008@gmail.com) and [sm.ahmadi@stu.qom.ac.ir](mailto:sm.ahmadi@stu.qom.ac.ir).



**Rouhollah Dianat**    was born in Qom, Iran, in 1977. He received his B.Sc. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2001, and his M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, in 2004. In 2010, he successfully completed his Ph.D. degree in the Department of Computer Engineering at the same institution. His research interests lie in the areas of signal processing, speech processing, image processing, and pattern recognition. He focuses on multi (hyper) spectral image analysis, pattern recognition and texture segmentation and classification. Dr. Rouhollah Dianat is currently an assistant professor in the Department of Computer Engineering and Information Technology at Qom University. He can be contacted at email: [rdianat@qom.ac.ir](mailto:rdianat@qom.ac.ir).