IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 2, April, 2024, pp. 949-959
Journal homepage: http://iieta.org/journals/ts

SNResNet: A New Architecture Based on SqNxt Blocks and Rish Activation for Efficient Face Recognition

Mostafa Diba, Hossein Khosravi*

Electronics - Image Processing, Faculty of Electrical Engineering, Shahrood University of Technology, Daneshgah Blvd., Shahrood 3619995161, Iran

Corresponding Author Email: hosseinkhosravi@shahroodut.ac.ir

https://doi.org/10.18280/ts.410235

Received: 31 July 2023
Revised: 20 September 2023
Accepted: 16 December 2023
Available online: 30 April 2024

Keywords:
deep learning, face recognition, Inception-ResNet, activation function, triplet loss, SqNxt block, ArcFace

ABSTRACT

In this paper, we present a novel face recognition architecture based on the Inception-ResNet framework, called SNResNet. The Inception-ResNet architecture, is effective in computer vision applications but exhibits limitations such as computational complexity, high memory consumption, and data dependency. It uses the ReLU activation function and softmax loss function which are not best-suited for face recognition. The proposed SNResNet uses triplet loss as the loss function to be able to train the model on large datasets. The advantages of the triplet loss over the softmax are handling one-shot learning, robustness to class imbalance and fine-grained discrimination. The ReLU activation function rejects all negative values that in some applications reduce the accuracy of the model. To overcome this problem, we introduced a new activation function called Rish which has better performance. In addition, we optimized the Inception-ResNet-B block using the SqNxt block to control the model's computational costs. The CASIA-WebFace dataset is used to train the models. This dataset has some challenges; e.g., some photos have more than one face, and all faces have a background. Preprocessing conditions are defined to identify and align the correct face. SNResNet achieves 94.63% accuracy on CASIA-WebFace. Performance evaluation on the LFW benchmark database yields an impressive accuracy of 99.68%, surpassing the standard model's accuracy of 98.85%. Further, we reduced the FLOPS of the Inception-ResNet model by 15.61% which indicates a lower computational cost and a faster model for face recognition.

1. INTRODUCTION

Face recognition is a biometric authentication method with many applications in the fields of public security, military, and attendance systems [1]. It can be implemented anywhere because it does not require complex hardware to capture data; we can install a camera and use facial recognition systems. However, other biometric authentication methods, such as iris and fingerprint detection, require special sensors that cannot be used everywhere. Face recognition holds a distinctive position among biometric authentication methods due to several key advantages. Unlike invasive methods such as iris or fingerprint recognition, face recognition is non-invasive and user-friendly. Its non-contact nature, which does not require physical interaction, contributes to user convenience and acceptance. This advantage aligns with human familiarity in recognizing faces, making it an intuitively accepted method. In addition, facial recognition is more affordable compared to other methods such as iris recognition that require special hardware. Its scalability allows for simultaneous recognition of multiple individuals, suited for crowd applications like surveillance. The versatility of integrating facial recognition into existing devices with cameras extends its usability. Face recognition stands out for its resistance to forgery, as deceiving robust face recognition systems is inherently challenging. This is especially true when advanced techniques take into account multiple facial features, ensuring higher security. Additionally, the adaptability of facial recognition to natural aging and appearance changes, such as hairstyles or facial hair, makes it a practical choice.

Thus, the combination of accessibility, versatility, security and compatibility of facial recognition makes it a compelling option for biometric authentication in various applications. For this reason, the development of facial recognition systems has always been one of the areas of interest among researchers.

Following the introduction of deep neural networks, large-scale facial recognition systems have been developed. Face recognition on a large scale has several challenges, including image distortion, racial changes, facial poses, expression, gender, and age changes. To overcome these challenges, it was necessary to design models that could extract appropriate features from data sets with low intra-class and high inter-class correlation to increase the generalization power of the models. Fortunately, in recent years, with the introduction of various deep architectures, many problems of face recognition systems have been solved [2-4]. Tran et al. [2] proposed an attendance kit, which integrates real-time Ultra-High Frequency (UHF) RFID technology with face recognition within a suite of

mobile applications tailored for institutions, lecturers, parents, and students. This tool is powered by the FaceNet model [5], renowned for its innovations, including the introduction of the triplet loss function, utilization of the Siamese Network Architecture, and implementation of triplet Selection Strategies. FaceNet is designed to withstand various facial changes, such as alterations in pose, lighting conditions, and facial expressions. This robustness is achieved through a combination of data augmentation techniques employed during training and the application of the aforementioned innovations.

In a survey paper, Wang and Deng [4] provide a comprehensive review of various approaches to large-scale face recognition. Domain adaptation methods play a crucial role in transferring knowledge from controlled environments to real-world settings. Transfer learning, on the other hand, harnesses pre-trained models to boost recognition performance. Ensemble learning combines predictions from multiple models, elevating accuracy and resilience. Techniques like GANs are employed for synthetic face generation, augmenting training data and fortifying resistance to age-related changes.

Researchers usually propose methods to increase the depth and width of models in order to solve complex problems. Increasing the depth means increasing the number of network layers, and increasing the width means increasing the number of blocks and filters in each layer. In a deep learning model, the higher the number of layers, the more abstract features the model can extract. The reason is the use of a non-linear activation function in each layer. Suppose we remove the activation function from a deep architecture with any number of layers. In that case, we will finally have a linear classifier, so the essential element in a deep learning model is its activation function. The activation function plays a fundamental role in extracting features, and the convergence speed of the model is affected by it.

In this paper, we optimize and use one of the favored architectures called Inception-ResNet for face recognition. The Inception-ResNet architecture stands out as a robust solution for face recognition tasks, offering several distinctive advantages over other deep learning architectures. It combines the strengths of Inception and ResNet, resulting in an efficient yet deep network that excels in capturing intricate facial features. This deep architecture addresses the vanishing gradient problem, enabling the learning of complex facial representations, particularly when dealing with extensive datasets. Inception-ResNet's hierarchical feature learning encompasses a wide range of facial characteristics, from low-level textures to high-level facial features, enabling it to excel in capturing the subtleties of facial identity. Its parallel processing capabilities, facilitated by Inception modules, empower the model to simultaneously consider both fine-grained and coarse facial details. Its ability to generalize well to diverse facial variations, such as pose, expression, and lighting, renders it suitable for real-world face recognition applications. According to the reasons that were raised, we chose the Inception-ResNet architecture as the basic architecture in our proposed model.

The optimization of this architecture consists of two parts. First, we define a new activation function called Rish and then generalize it to obtain two models using the Rish activation function and its generalization to achieve higher accuracy. The second novelty is fusing the Inception-ResNet-B block of this architecture with the SqNxt block [6]; the idea is to reduce the computational cost of the model. We trained the proposed models with the CASIA-WebFace dataset, and evaluated their performance with the LFW benchmark [7, 8]. The results showed that the FLOPS (floating-point operations per second) of the proposed models are 2.4013B and 2.4042B, respectively, while the FLOPS of the Inception-ResNet architecture is 2.8492B. Also, the proposed models achieved the accuracies of 99.41% and 99.68% on the LFW benchmark which are 0.36% and 0.63%, better compared to the initial model. In other words, we were able to increase the accuracy of the model using proposed activation functions and reduce the computational cost using the Inception-ResNet-B block along with SqNxt. The rest of the paper is organized as follows: In section 2, the activation functions of deep neural networks will be reviewed.

section 3 presents the proposed activation functions. The CASIA-WebFace dataset, its challenges and proposed solutions are discussed in section 4. In section 5, we will briefly introduce the Inception-ResNet architecture and detail the modifications we have made to this architecture. The experimental results will be expressed in section 6. Sections 7 and 8 are devoted to the discussion and conclusion, respectively.

2. HISTORY OF ACTIVATION FUNCTIONS

As mentioned in the previous section, the activation function is one of the essential elements in deep learning models. In this section, we will review different types of activation functions, and will state the advantages and disadvantages of them.

- **Sigmoid**

The sigmoid function has been the most widely used in neural networks. As seen in Figure 1, the output of this function is always in the range [0, 1]. For this reason, this function does not have a zero mean, which is one of the disadvantages of this activation function. Eq. (1) expresses the sigmoid function.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

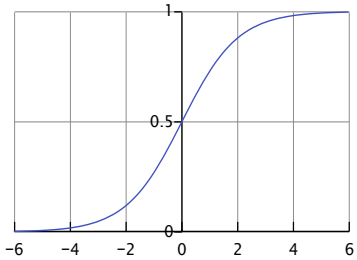

Figure 1. Sigmoid function

Among other disadvantages of this function, we can mention saturated areas. As shown in Figure 1, the function's behavior is linear in areas close to zero. However, the function's gradient is shallow when we move away from zero, for example, for data greater than *+4* or smaller than *-4*, which indicates that the gradient of this function tends to be zero in these areas. This problem causes the network weights not to be updated during the network training in the backpropagation

stage. In addition, there is an exponential function (e^x) in the Sigmoid function, which increases the computational cost and reduces the model training speed.

• **Tanh**

The output of the Tanh activation function is in the interval [-1,1], which gives it a zero mean. This is an advantage of the tanh activation function over the sigmoid. As shown in Figure 2, the tanh activation function has saturation areas like the sigmoid, and the same problem of zero gradients is also present in this function.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2}$$

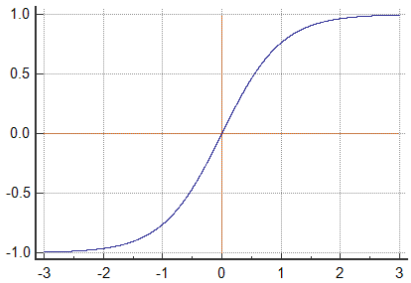

Figure 2. Tanh activation function

• **ReLU**

The ReLU activation function introduced in 2012 by Krizhevsky et al. [9] is a piecewise linear function. As seen in Figure 3, in this function, the output is zero for negative data, and equal to the input value for positive data.

$$ReLu = \max(0, x) \tag{3}$$

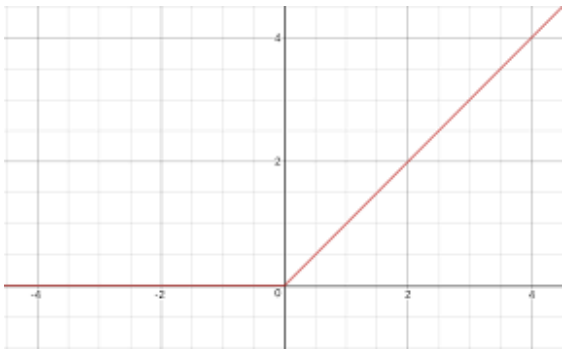

Figure 3. ReLU activation function

Compared to the previous functions, the advantages of ReLU activation function are as follows:

• It does not saturate for positive data, and the gradient is always one in these areas.

• It is very computationally efficient: Sigmoid and tanh functions require exponentiation (e^x), which increases the computational cost, while the ReLU function does not.

• Reports have indicated that using the ReLU activation function in a deep architecture makes the model converge up to 6 times faster than tanh and sigmoid.

The above advantages have made the ReLU activation function one of the most popular activation functions introduced so far, and most deep learning models use this function by default. However, ReLU also has disadvantages, that include:

• The function is not continuous or differentiable for data with zero value. This challenge is unimportant because we rarely have data whose value is zero.

• ReLU does not have zero mean.

• Dead ReLU: The function's gradient is zero for negative inputs, so the network is not trained in this region in the backpropagation stage. Also, due to the initial weighting, several filters may not extract any features at the beginning of the training and only impose an additional computational cost on the model.

• **Leaky ReLU**

As mentioned, one of the challenges that ReLU faces is the zero output for negative data. To solve this problem, Mass et al. introduced the Leaky ReLU activation function [10]. Which considered a small slope (for example, 0.01) for the negative area (Eq. 4) Figure 4 shows the plot of Leaky ReLU.

$$f(x) = \max(0.01x, x) \tag{4}$$

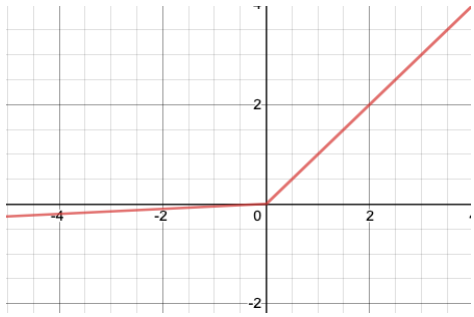

Figure 4. Leaky ReLU activation function

The problem of dead filters was solved using this method, and the average of this function was closer to zero. The gradient of this function for negative data is a small constant we have considered (for example, 0.01). The question arises what should be the constant coefficient considered for the negative area, and what is the best coefficient? The ideal coefficient will differ for different problems, so proposing a fixed number to solve all problems is impossible. Researchers introduced the PReLU activation function to solve this challenge.

• **Parametric ReLU (PReLU)**

He et al. [11] introduced the PReLU activation function in 2015. The gradient control coefficient of the negative part is determined as a hyper-parameter, and the model should obtain the appropriate value through training in the backpropagation stage (Eq. (5) and Figure 5).

$$f(x_i) = \max(0, x_i) + a_i \min(0, x_i) \tag{5}$$

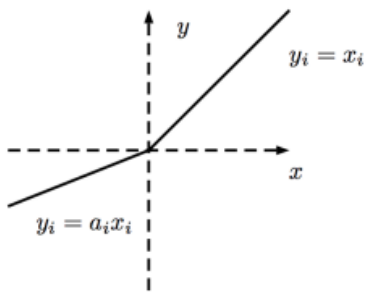

Figure 5. PReLU activation function

• **ELU**

Clevert et al. [12] introduced another modification of ReLU using an exponential function for negative regions to make the function smoother (Figure 6). This function called ELU [12] behaves more smoothly than PReLU, but due to the exponential term for negative area (Eq. (6)); It increases the computational cost of the model.

$$\begin{cases} x & x > 0 \\ \alpha(exp(x) - 1) & x \leq 0 \end{cases} \qquad \alpha > 0 \tag{6}$$

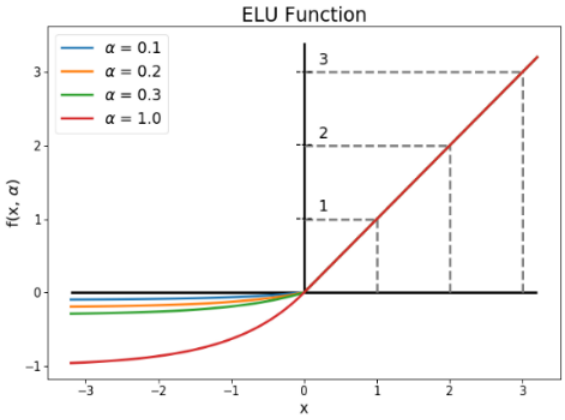

Figure 6. ELU activation function

• **Softplus**

Zheng et al. [13] used the Softplus function as an activation function for the first time. This function is known as the softened ReLU (Figure 7). The advantage of this function compared to the ReLU is its soft derivative, which is helpful in the backpropagation stage. This function has a higher computational cost than ReLU due to exponential and log terms (Eq. (7)). The classification accuracy of Softplus is 5.81% lower than that of ReLU, and the average training time of the model that uses Softplus is 135 seconds more than the model that uses ReLU [14]. For this reason, using Softplus as an activation function in deep learning models is not recommended.

$$f(x) = log(1 + e^x) \tag{7}$$

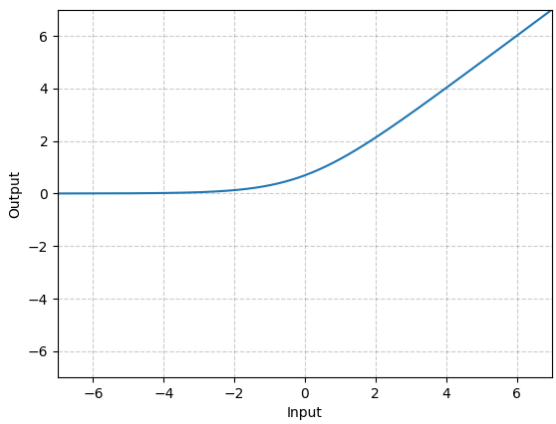

Figure 7. Softplus activation function

• **Swish**

Google researchers introduced the Swish activation function in 2017 as an alternative to ReLU [15]. This function has a trainable hyperparameter called β (Eq. (8)). The behavior of this non-linear function varies smoothly from linear to ReLU-like based on the value of the hyperparameter β. Figure 8 shows the graph of the Swish function for different values of β.

$$f(x) = x.sigmoid(\beta x) \tag{8}$$

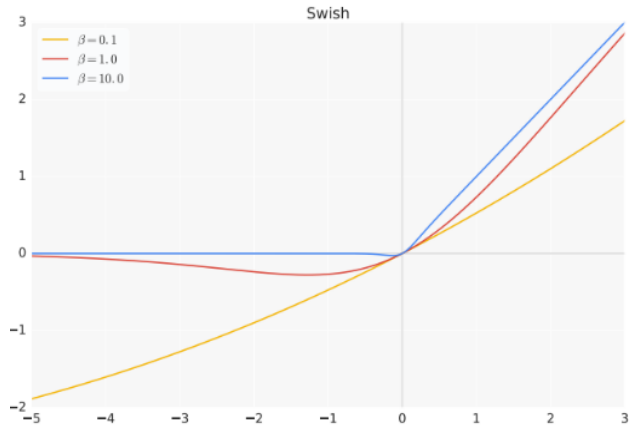

Figure 8. Swish activation function, the behavior of this function changes smoothly from a linear function to a non-linear function for different values of β

This function has almost zero mean for small values of β. Because the Swish activation function is continuous, its gradient can be calculated for any input value, and the gradient vanishing challenge does not exist for negative values. However, its computational cost is more than ReLU.

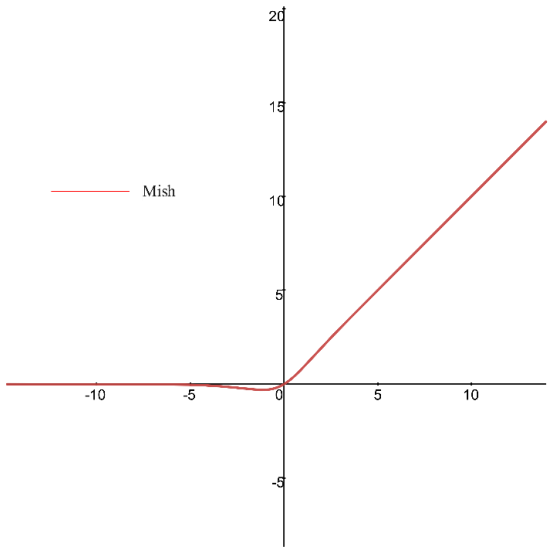

Figure 9. Mish activation function

• **Mish**

Mish is One of the newest activation functions introduced in 2019 by Mish [16] (Figure 9). According to the mish equation (Eq. (9)), its computational cost is significantly higher than that of ReLU. For this reason, the run time of the models implemented with mish is higher than those implemented with ReLU. For example, the execution time in the forward stage for float16 data in the model implemented with ReLU is 223.7 microseconds. In contrast, if the same

model uses the mish activation function, the run time in the forward stage will be 658.8 microseconds [16]. However, the accuracy obtained in mish is higher than other activation functions introduced. Therefore, using this activation function is recommended where accuracy is important.

$$\begin{cases} f(x) = x.tanh(softplus(x) \\ softplus(x) = log(1 + e^x) \end{cases} \quad (9)$$

3. PROPOSED ACTIVATION FUNCTION - RISH

As discussed earlier, the activation function is one of the essential elements in deep architecture. If the activation function is selected correctly, the accuracy and convergence speed of the model will increase. For this reason, one of our goals in this article is to design a new activation function. As stated, ReLU is one of the best functions ever introduced as an activation function. Although ReLU is not computationally complex, it performs well for positive data. Because the ReLU gradient for positive data is one, and in the backpropagation stage, the gradients obtained from the previous layer, transferred to the next layer without vanishing. However, it faces challenges for negative data; this has caused new activation functions to develop. The Mish activation function ranks among the most well-designed activation functions ever created. Mish addresses the " dead ReLU " problem, reducing inactive neurons during training, thus enhancing learning capacity. Unlike ReLU, which zeros negative values, Mish offers a smooth transition for both positive and negative inputs, preserving information from negative activations, and making it advantageous for data with negative values. Unfortunately, as stated in section 2, Mish involves a more complex mathematical formulation compared to the ReLU. This complexity increases computation time and resource requirements.

As you can see in Figure 10, in the proposed activation function, the output is defined as ReLU, for positive data, and Mish for negative data. we are removing the computational complexity of the Mish activation function for positive data, thus reducing the computational cost of the model. On the other hand, negative data is treated like Mish, which has higher accuracy among other activation functions. We call our proposed activation function as Rish (ReLU + Mish), which is defined in Eq. (10):

$$f(x) = \begin{cases} x & x \geq 0 \\ x.\tanh\left(softplus(x)\right) & x < 0 \end{cases}$$
Or:
$$f(x) = max(0, x) + min(0, x.tanh(1 + e^x)) \quad (10)$$

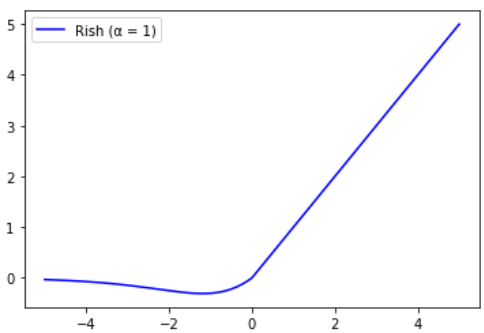

Figure 10. Proposed activation function, called Rish

The Rish activation function can also be generalized and had a trainable hyperparameter α as in Eq. (11):

$$f(x) = \begin{cases} x & x \geq 0 \\ x.\tanh\left(softplus(\alpha x)\right) & x < 0 \end{cases}$$
Or:
$$f(x) = max(0, x) + min(0, x.tanh(1 + e^{\alpha x})) \quad (11)$$

The best choice of hyperparameter α for the proposed model was found to be 0.01. We employed the random search method to determine the optimal value of this hyperparameter, with the objective of mitigating the adverse impact of negative data during model training. To achieve this, we constrained the α value to be less than 0.5. Under this constraint, we defined a reasonable range for the hyperparameter α, encompassing values such as 0.001, 0.005, 0.01, 0.02, and so forth, up to 0.5. To assess the effectiveness of different α values, we conducted experiments on a subset of 100 classes from the CASIA-WebFace training dataset. Each α value was evaluated by training our model over five iterations. We employed accuracy as the primary criterion for assessing the optimality of α. As a result of our evaluation process, we identified the optimal value for the hyperparameter α to be 0.01. Figure 11 shows the graph of Eq. (11) for α=0.01.

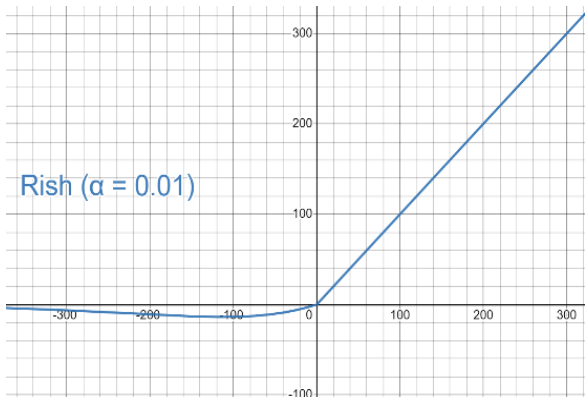

Figure 11. Graph of the generalized Rish activation function, for α=0.01

To validate our assertion that the utilization of the Rish activation function leads to a reduction in computational costs, we conducted an analysis of runtime performance during the forward pass. To facilitate this assessment, we constructed a simple model comprising two convolutional layers. In this model, we initially employed the Mish activation function and calculated the mean execution time over 100 iterations using float16 data. Subsequently, we substituted the activation function with Rish. Our observations revealed that the average execution times for the models were 0.126357 and 0.109018 seconds, respectively. Notably, the model employing the Rish activation function exhibited a 14.73% improvement in average execution speed compared to the one utilizing the Mish activation function.

4. CASIA-WEBFACE DATASET AND MTCNN FACE DETECTOR

The training dataset used to train the face recognition models proposed in this article is CASIA-WebFace. This dataset contains 494,414 images in 10,575 classes. Figure 12 displays some samples of this dataset.

Figure 12. Sample data from the CASIA-WebFace dataset

Figure 12 shows that the CASIA-WebFace images are not focused on people's faces, and each image contains a part of the body and the background in addition to the person's face. As a result, the first stage of pre-processing is the extraction of people's faces from the images. For this purpose, we used the MTCNN face detector [17]. Another challenge is that there is more than one face in some images of the dataset (Figure 12). This challenge causes more than one face to be extracted for some images during the face detection by MTCNN. It will increase the label noise of the dataset. To solve this problem, we made two assumptions:

(a) In every image with more than one face, we choose the face with a higher confidence as the main face.

(b) If two faces have the same confidence, we assume that the image of the main face is always bigger than the image of the side face because the purpose of collecting this image was the first person. With this assumption, we extract the image of a person with larger dimensions as the target image.

Figure 13 shows this challenge and how to solve it.

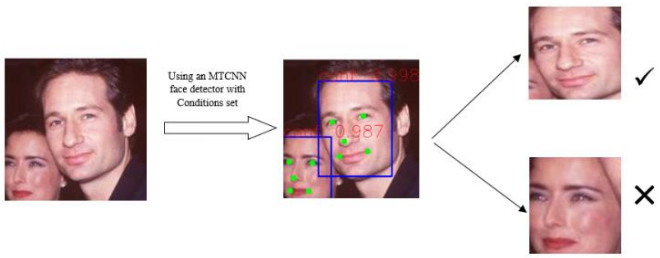

Figure 13. Data preprocessing using an MTCNN face detector in order to extract the main face from each image

In order to reduce the computational cost of the face recognition model, we considered the size of the input images to be 160×160 pixels. LFW benchmark is used to check the performance of the final model.

5. PROPOSED FACE RECOGNITION MODEL

As mentioned in section 4, we first extract the target face from the images in the CASIA-WebFace dataset. Now, convenient features for face recognition should be extracted using a deep neural network. For this purpose, we optimized the Inception-ResNet and used it to extract features. Figure 14 displays the basic architecture of this network in standard mode [18].

As shown in Figure 14, the size of input images for standard Inception-ResNet is considered (229×229×3). To reduce the computational cost of the model, we set the dimensions of the input images to (160×160×3). Figure 15 displays one of the blocks used in this architecture, called Inception-ResNet-B.

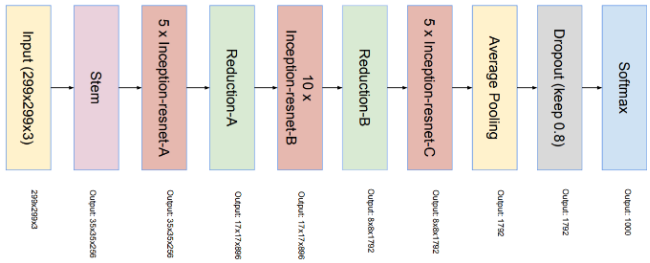

Figure 14. Inception-ResNet standard architecture [18]

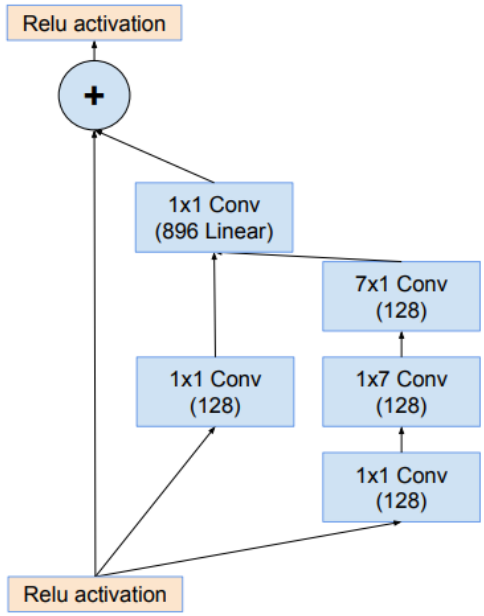

Figure 15. Inception-ResNet-B block [18]

To reduce the computational cost, we replaced the third branch of the Inception-ResNet-B block with the SqueezeNext (SqNxt) block and named the new block SNResNet. In 2018, Gholami et al. designed the SqNxt architecture to reduce the computational cost of models implemented on embedded systems. This block consists of a combination of ResNet and SqueezeNet blocks and their optimization [6].

The SqNxt block presents a set of noteworthy advantages that position it as a valuable component within deep neural network architectures. It adeptly strikes a balance between model size and computational efficiency, a critical consideration in contemporary deep learning endeavors. This balance is elegantly achieved by the incorporation of a "squeeze" layer, which harmoniously combines 1x1 convolutions and pointwise activations. This amalgamation serves the purpose of effectively reducing input channels while preserving essential information. In addition, the ability of the SqNxt architecture is highlighted by the presence of expand layers. These layers harness the potential of 1x1 convolutions, followed by 3x3 convolutions, yielding a judicious augmentation of channel dimensions. This augmentation significantly strengthens the model's capacity to represent complex features without imposing undue computational burdens. Additionally, SqNxt introduces skip connections, drawing inspiration from the ResNet paradigm. These connections are instrumental in facilitating more streamlined gradient flow during training, thus enhancing the model's training efficiency and convergence properties. In pursuit of further computational efficiency, certain iterations

of SqNxt incorporate reduction layers. These layers judiciously reduce spatial dimensions within feature maps, thereby yielding substantial reductions in computational overhead for subsequent layers. Collectively, the SqNxt block exemplifies a refined approach to deep learning, enabling the creation of compact yet potent models ideally suited for deployment in resource-constrained settings or applications necessitating real-time processing. Figure16 shows the structure of the SqNxt block.

Figure 17 shows the proposed SNResNet block, an optimized version of the Inception-ResNet-B block using the SqNxt block.

With this method, the number of computational parameters of the model decreased from 22,779,312 to 19,258,384. This amount is equivalent to a 15.56% reduction in the calculation parameters of the model. The next step to optimize the basic architecture of the Inception-ResNet is to use the activation functions proposed in section 3. Inception-ResNet, by default, uses the ReLU activation function. At this stage, we replaced ReLU with our proposed activation functions. We will finally have two models using the proposed activation function and its generalized version. The first model uses the activation function of Eq. (10), and the other uses the activation function of Eq. (11). Figure 18 shows the final architecture of the proposed SNResNet block after replacing the proposed activation functions.

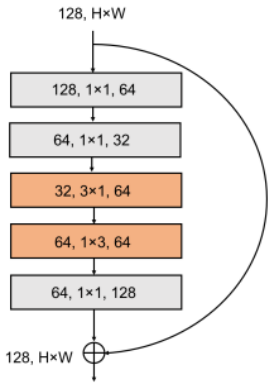

Figure 16. SqNxt block structure [18]

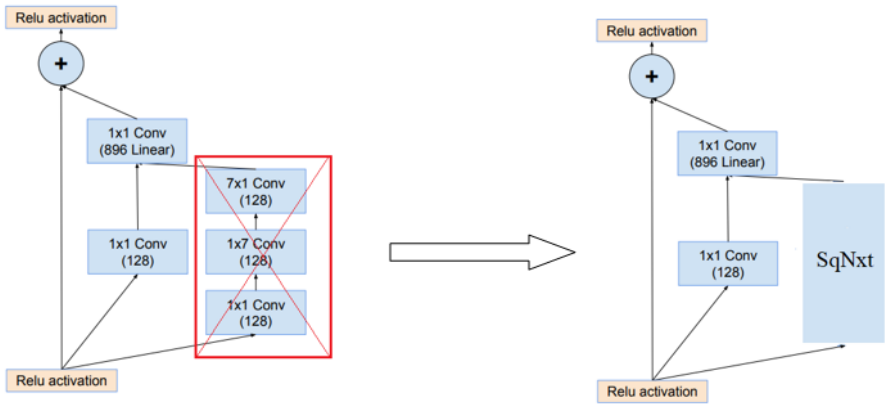

Figure 17. An optimized version of Inception-ResNet block-B by using SqNxt block

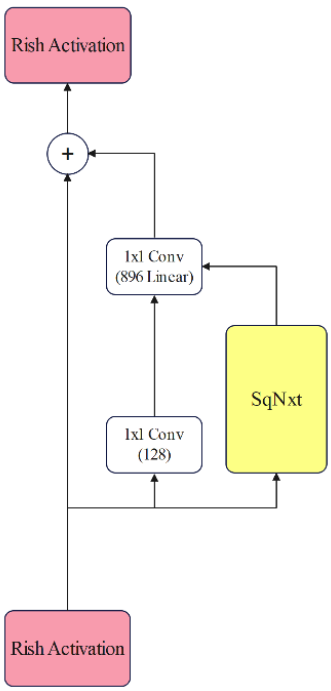

Figure 18. Optimized Inception-ResNet-B block using SqNxt block and suggested activation functions

In the training phase, the triplet loss is used as our loss function. This function was first introduced by the Google research team to design the FaceNet model [5]. We also used the Adam optimizer to optimize the model in the backpropagation stage [19]. Our models extract 512 features from people's faces, by which each person can be identified.

6. EXPERIMENTAL RESULTS

In this section, we will express the experimental results obtained by the proposed models. We used the CASIA-WebFace dataset to train the models. Model's batch size in the training stage was set to 8. The training process was conducted as follows: first, face detection is performed by the method presented in section 4. Then the according to the requirement of triplet loss, three faces (anchor, positive and negative) are entered into the model for training in groups of 8. Figure 19 shows an example batch of the input data to the model.

Each model is trained for 50 epochs. The first model is implemented with the Rish activation function (Eq. (10) and α=1) and the second with α = 0.01. The first model obtained the accuracy of 91.95%, and the second achieved the higher accuracy of 94.63%. Figures 20 and 21 show the training graphs for the two proposed models, respectively.

Figure 19. A batch of face images prepared for training the model

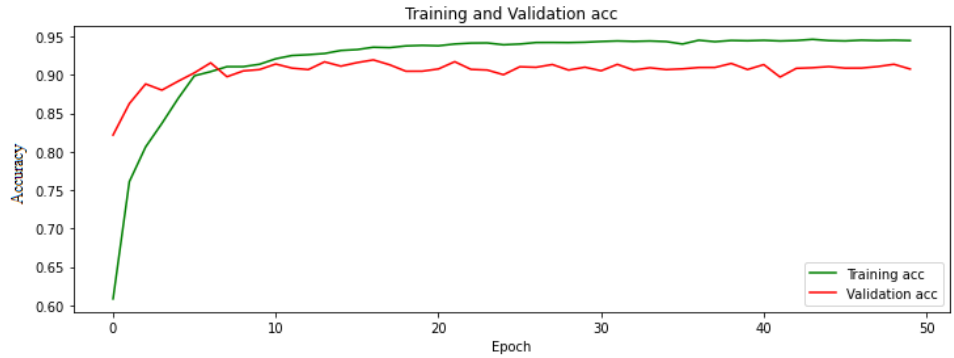

Figure 20. Accuracy graph of training and validation data for the first model (The activation function is Rish with α=1)

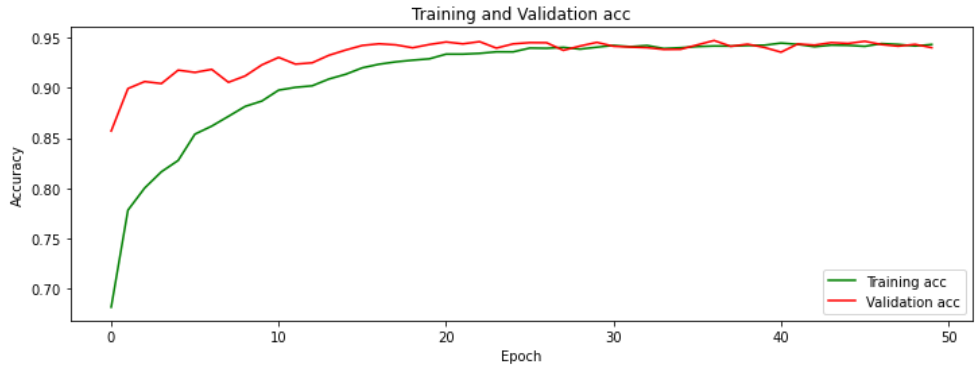

Figure 21. Accuracy graph of training and validation data for the second model (The activation function is Rish with α=0.01)

In Table 1, we compared the accuracy of the validation data of the proposed models with other models. We compare the Inception-ResNet network with different activation functions against the optimized SNResNet network.

Table 1. Comparing the accuracy of the proposed models with other models

Architecture	Activation Function	Loss Function	Accuracy
Inception-ResNet	ReLU	ArcFace [20]	84.24%
Inception-ResNet	RReLU	ArcFace	83.75%
Inception-ResNet	SELU	ArcFace	80.62%
Inception-ResNet	Mish	ArcFace	84.41%
Inception-ResNet	**Rish (α=1)**	ArcFace	84.96%
SNResNet	**Rish (α=1)**	Triplet Loss	**91.95%**
SNResNet	**Rish (α=0.01)**	Triplet Loss	**94.63%**

Figure 22. Some results of the validation data that shows the resistance of the proposed models against age changes

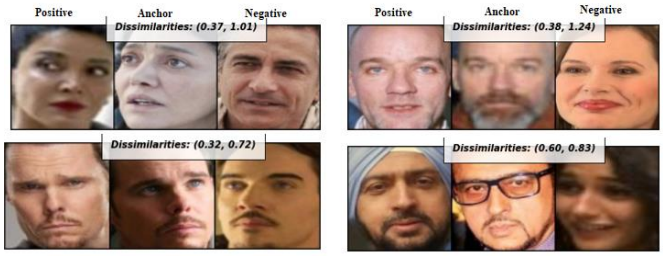

Figure 23. Some results of the validation data that show the resistance of the proposed models to changes in facial appearance

Figure 24. Some results of the validation data that shows the resistance of the proposed models against emotional changes

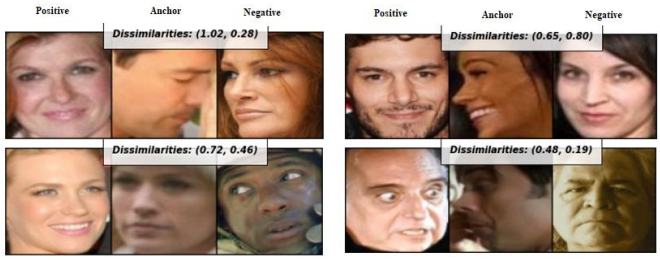

Figure 25. An example of validation data that includes label noise (the positive sample is not the same as the anchor)

Our proposed models have better generalization power. As shown in Figure 22, the proposed models resist against age changes.

In addition, the proposed model of SNResNet was able to withstand facial appearance changes, such as makeup, beard, glasses, and other things. Figure 23 shows this issue.

The results of the validation data show that the proposed model is also resistant to emotional changes like anger, sadness, and happiness, examples of which are shown in Figure 24.

Unfortunately, the CASIA-WebFace dataset has label noise. Hu et al. reported that this noise is between 9.3% and 13% [21]. This means that when we make our triple data, there is a possibility that this noise will cause the model to be wrong. We observed an average of 4.75% label noise on 100 samples of validation data outputs. To check the amount of label noise, we obtained label noises four times, each time in 100 validation data, in which 3, 5, 5, and 6 label noises were observed, respectively. As a result, in ideal conditions, any model trained with this data set will have about a 5% error in the validation data. So, for the models trained with the Casia-WebFace dataset, the maximum accuracy they can achieve in the validation stage is 95%.

In Figures 22-25 the first number indicates the distance between the anchor and the positive, and the second indicates the distance between the anchor and the negative.

Figure 25 displays an example of a label noise challenge.

We investigated the FLOPS of the proposed models next. The obtained FLOPS for our models are 2.4013B and 2.4042B, respectively ("B" stands for billion). These values of FLOPS are 15.72% and 15.61% lower than the standard model, respectively, while our proposed activation function had more computational cost; this indicates that optimizing the Inception-ResNet-B block by the SqNxt block has been efficient. Table 2 compares the amount of FLOPS of our models with other models. Our models show the least amount of FLOPS in this table.

Table 2. Comparison of FLOPS of the proposed models with standard models. (B stands for billion)

Backbone	Activation Function	FLOPS
Inception-ResNet-V1	ReLU	2.8492B
Inception-ResNet-V1	Mish	2.8523B
SNResNet (proposed)	**Rish (α=1)**	**2.4013B**
SNResNet (proposed)	**Rish (α=0.01)**	**2.4042B**

Table 3. Comparing the classification accuracy of the proposed models with other advanced models on LFW dataset

Method	Year	Loss	Architecture	Training Set	Accuracy
SphereFace [22]	2018	A-Softmax	ResNet64	CASIA-WebFace	99.42%
	2019	AMS-Softmax	ResNet50	CASIA-WebFace	99.34% [23]
Marginal Loss	2019	Marginal Loss	ResNet50	CASIA-WebFace	98.91% [23]
ArcFace	2019	ArcFace	ResNet50	CASIA-WebFace	99.35% [23]
ACNN [24]	2020	Arcface	ResNet-100	DeepGlint-MS1M	99.83
RCM Loss [25]	2020	Rotation Consistent Margin loss	ResNet-18	CASIA-WebFace	98.91
Ben Fredj et al.'s work [26]	2021	Softmax with center loss	GoogleNet-Inception	CASIA-WebFace	99.2
-----	2023	Softmax	Inception-ResNet-V1	CASIA-WebFace	98.85%
-----	2023	Triplet Loss	Inception-ResNet-V1	CASIA-WebFace	99.05%
IAM [27]	2020	IAM loss	Inception ResNet-V1	CASIA-WebFace	99.12
SNResNet	2023	Triplet Loss	**SNResNet (Rish (α=1))**	**CASIA-WebFace**	**99.41%**
SNResNet	2023	Triplet Loss	**SNResNet(Rish (α=0.01))**	**CASIA-WebFace**	**99.68%**

We used the LFW dataset to check the performance of the proposed models with other advanced face recognition models. Table 3 compares the classification accuracy of the proposed models with other face recognition models using LFW dataset. The classification accuracy of our models was 99.41% and 99.68%, respectively.

7. DISCUSSION

During this work, we encountered certain limitations, notably related to hardware constraints during the model training process. Specifically, we used a 1660 GPU for the training, which imposed a restriction on the achievable batch size, capping it at 8.

Another problem was label noise of the CASIA-WebFace dataset. As indicated in Section 6, this dataset contains various types of noise, with approximately 5% attributed to label noise among the overall range of 9.3% to 13%. Regrettably, the task of identifying and mitigating this noise significantly escalates the computational demands on our system, and rectifying this dataset's label noise presents a formidable challenge that could serve as a potential avenue for future research. This label noise presence had a notable impact on the accuracy of all models trained on this dataset, none of which achieved an accuracy exceeding 90% during the evaluation phase with test data. However, our proposed model managed to attain an impressive accuracy of 94.63%, marking the highest reported accuracy to date.

The dataset primarily comprises classes related to Hollywood celebrities, resulting in limited ethnic diversity representation. This inherent challenge can hinder the model's performance across various ethnicities. To address this issue, we applied data augmentation techniques.

The CASIA-WebFace dataset, like many face recognition datasets, exhibits a long-tail distribution in terms of the number of images per individual or identity. In a long-tail distribution, a small number of identities have a large number of images, while the majority of identities have relatively few images. This distribution is common in real-world scenarios, as there are typically many more "common" individuals with numerous images (e.g., celebrities or public figures) and relatively fewer images of "uncommon" individuals. The long-tail distribution in the CASIA-WebFace dataset can have important implications for face recognition research and the training of machine learning models. When the dataset is used to train models, the overrepresentation of some identities and underrepresentation of others can lead to issues like bias and overfitting. Models may perform exceptionally well on the well-represented identities but struggle with the less-represented ones. To mitigate the effects of the long-tail distribution, we employed data augmentation techniques and a triplet loss function.

8. CONCLUSIONS

In this paper, we proposed a face recognition model to enhance face identification accuracy and decrease the model's computing cost. For this purpose, we introduced a new deep architecture called SNResNet. The proposed model is an optimized deep architecture of Inception-ResNet. In SNResNet, we combined the Inception-ResNet-B block with the SqNxt block. Using this method, we reduced the FLOPS of the proposed models by 15.72% and 15.61%, respectively, compared to the standard model. Furthermore, the number of model parameters optimized by our proposed method is 15.56% less than the standard model. To increase the accuracy of the proposed model, we introduced a new activation function called Rish which is a combination of ReLU and Mish. Then we designed the generalized Rish activation function with a trainable hyper-parameter to optimize the Rish activation function to solve different problems. We used the Rish activation function and its generalization in the SNResNet architecture and finally obtained two face recognition models. We considered triplet loss as the loss function for both models and used Adam as the optimizer. We used the CASIA-WebFace dataset to train the proposed models. The second model, Rish with α=0.01, achieved an accuracy of 94.63% for the validation data using the Casia-WebFace dataset. However, the highest possible accuracy for this dataset is 95% in ideal conditions. To check the performance of the proposed models and compare them with other models, we used the benchmark data set of LFW. The classification accuracy in the LFW criterion for our proposed models are 99.41% and 99.68%, respectively. In comparison, the model's accuracy used the Inception-ResNet base network, and the triplet loss function is 99.05%; This shows the excellent performance of Rish's activation function.

In addition to face recognition, SNResNet works well in other tasks like image classification, object detection, semantic segmentation, medical image analysis as well as Inception-ResNet architecture. Because SNResNet optimized Inception-ResNet architecture. Therefore, the use of SNResNet in these tasks is recommended as future works.

SNResNet can work well for practical face recognition applications such as surveillance systems, identity verification and authentication, healthcare, airports and travel. Because in practical applications, model accuracy and model size are very important. Our model with 94.63% accuracy and its 90 MB size is efficient for implementation on most of todays' computers.

REPLICATION OF RESULTS

The code of the paper is placed in the link below:
https://github.com/mosdiba/SNResNet/blob/4e5d9ed6925eb38ab8ce9eb3b8b4cd80fffbc7e2/SNResNet.ipynb.

The dataset used in this article is CASIA-WebFace, which could not be uploaded due to its large volume.

REFERENCES

[1] Smith, M., Miller, S. (2022). The ethical application of biometric facial recognition technology. Ai & Society, 1-9. https://doi.org/10.1007/s00146-021-01199-9

[2] Tran, T.D., Huynh, K.T., Nguyen, P.Q., Ly, T.N. (2022). AttendanceKit: A set of role-based mobile applications for automatic attendance checking with UHF RFID using Realtime firebase and face recognition. In: Dang, T.K., Küng, J., Chung, T.M. (eds) Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications. FDSE 2022. Communications in Computer and Information Science, vol 1688. Springer, Singapore. https://doi.org/10.1007/978-981-19-8069-5_29

[3] Waelen, R.A. (2023). The struggle for recognition in the age of facial recognition technology. AI and Ethics, 3(1): 215-222. https://doi.org/10.1007/s43681-022-00146-8
[4] Wang, M., Deng, W. (2021). Deep face recognition: A survey. Neurocomputing, 429: 215-244. https://doi.org/10.1016/j.neucom.2020.10.081
[5] Schroff, F., Kalenichenko, D., Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, pp. 815-823. https://doi.org/10.1109/CVPR.2015.7298682
[6] Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S.C., Keutzer, K. (2018). SqueezeNext: Hardware-aware neural network design. Neural and Evolutionary Computing. https://doi.org/10.48550/arXiv.1803.10615
[7] Yi, D., Lei, Z., Liao, S., Li, S.Z. (2014). Learning face representation from scratch. arXiv preprint arXiv:1411.7923. https://doi.org/10.48550/arXiv.1411.7923
[8] Huang, G.B., Learned-Miller, E. (2014). Labeled faces in the wild: Updates and new reporting procedures. Computer Science, Corpus ID: 17716267.
[9] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.
[10] Maas, A.L., Hannun, A.Y., Ng, A.Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia, USA.
[11] He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, pp. 1026-1034. https://doi.org/10.1109/ICCV.2015.123
[12] Clevert, D.A., Unterthiner, T., Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289. https://doi.org/10.48550/arXiv.1511.07289
[13] Zheng, H., Yang, Z., Liu, W., Liang, J., Li, Y. (2015). Improving deep neural networks using softplus units. In 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, pp. 1-4. https://doi.org/10.1109/IJCNN.2015.7280459
[14] Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks. Bio-Inspired Neurocomputing, 203-224. https://doi.org/10.1007/978-981-15-5495-7_11
[15] Ramachandran, P., Zoph, B., Le, Q.V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941. https://doi.org/10.48550/arXiv.1710.05941
[16] Mish, M.D. (2019). A self regularized non-monotonic activation function. arXiv preprint arXiv:1908.08681. https://doi.org/10.48550/arXiv.1908.08681
[17] Zhang, K., Zhang, Z., Li, Z., Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10): 1499-1503. https://doi.org/10.1109/LSP.2016.2603342
[18] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 31(1). https://doi.org/10.1609/aaai.v31i1.11231
[19] Kingma, D.P., Adam, J.B. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980
[20] Deng, J., Guo, J., Xue, N., Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, pp. 4690-4699. https://doi.org/10.1109/CVPR.2019.00482
[21] Hu, W., Huang, Y., Zhang, F., Li, R. (2019). Noise-tolerant paradigm for training face recognition CNNs. https://doi.org/10.48550/arXiv.1903.10357
[22] Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 6738-6746. https://doi.org/10.1109/CVPR.2017.713
[23] Srivastava, Y., Murali, V., Dubey, S.R. (2020). A performance evaluation of loss functions for deep face recognition. In Computer Vision, Pattern Recognition, Image Processing, and Graphics: 7th National Conference, NCVPRIPG 2019, Hubballi, India, pp. 322-332. https://doi.org/10.1007/978-981-15-8697-2_30
[24] Ling, H., Wu, J., Huang, J., Chen, J., Li, P. (2020). Attention-based convolutional neural network for deep face recognition. Multimedia Tools and Applications, 79: 5595-5616. https://doi.org/10.1007/s11042-019-08422-2
[25] Wu, Y., Wu, Y., Gong, R., Lv, Y., Chen, K., Liang, D., Hu, X.L., Liu, X.L., Yan, J. (2020). Rotation consistent margin loss for efficient low-bit face recognition. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, pp. 6866-6876. https://doi.org/10.1109/CVPR42600.2020.00690
[26] Ben Fredj, H., Bouguezzi, S., Souani, C. (2021). Face recognition in unconstrained environment with CNN. The Visual Computer, 37: 217-226. https://doi.org/10.1007/s00371-020-01794-9
[27] Sun, J., Yang, W., Gao, R., Xue, J.H., Liao, Q. (2020). Inter-class angular margin loss for face recognition. Signal Processing: Image Communication, 80: 115636. https://doi.org/10.1016/j.image.2019.115636