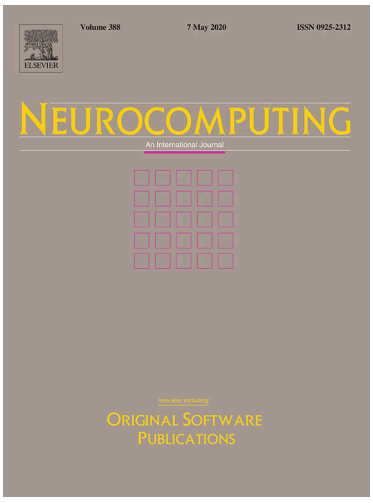
Journal Pre-proofs




# More Trainable Inception-ResNet for Face Recognition

Shuai Peng, Hongbo Huang, Weijun Chen, Liang Zhang, Weiwei Fang

PII: S0925-2312(20)30857-2
DOI: https://doi.org/10.1016/j.neucom.2020.05.022
Reference: NEUCOM 22339

To appear in: *Neurocomputing*

Received Date: 18 July 2019
Accepted Date: 14 May 2020

Please cite this article as: S. Peng, H. Huang, W. Chen, L. Zhang, W. Fang, More Trainable Inception-ResNet for Face Recognition, *Neurocomputing* (2020), doi: https://doi.org/10.1016/j.neucom.2020.05.022

# More Trainable Inception-ResNet for Face Recognition

Shuai Peng[a], Hongbo Huang[a,b,*], Weijun Chen[a], Liang Zhang[a], Weiwei Fang[a,b]

[a] *Computer School, Beijing Information Science and Technology University, Beijing 100192, China*
[b] *Institute of Computing Intelligence, Beijing Information Science and Technology University, Beijing 100192, China*

**Abstract**

In recent years, applications of face recognition have increased significantly. Despite the successful application of deep convolutional neural network (DCNN), training such networks is still a challenging task that needs a lot of experience and carefully tuning. Based on the Inception-ResNet network, we propose a novel method to mitigate the difficulty of training such deep convolutional neural network and improve its performance simultaneously. The residual scaling factor used in the Inception-ResNet module is a manually set fixed value. We believe that changing the value to a trainable parameter and initializing it to a small value can improve the stability of the model training. We further adopted a small trick of alternating the ReLU activation function with the Leaky ReLU and PReLU. The proposed model slightly increased the number of training parameters but improved training stability and performance significantly. Extensive experiments are conducted on VGGFace2, MS1MV2, IJBB and LFW datasets. The results show that the proposed trainable residual scaling factor (TRSF) and PReLU can promote the accuracy notably while stabilizing training process.

*Keywords:* deep learning, convolutional neural network, face recognition, Inception-ResNet network, activation function

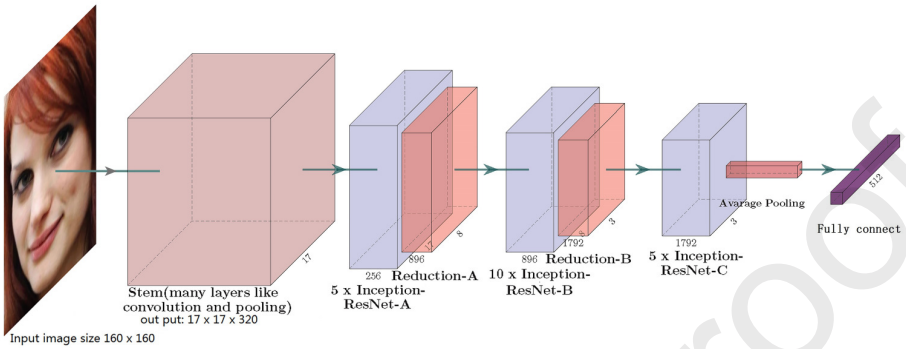
*Corresponding author
*Email address:* hhb@bistu.edu.cn (Hongbo Huang)

## 1. Introduction

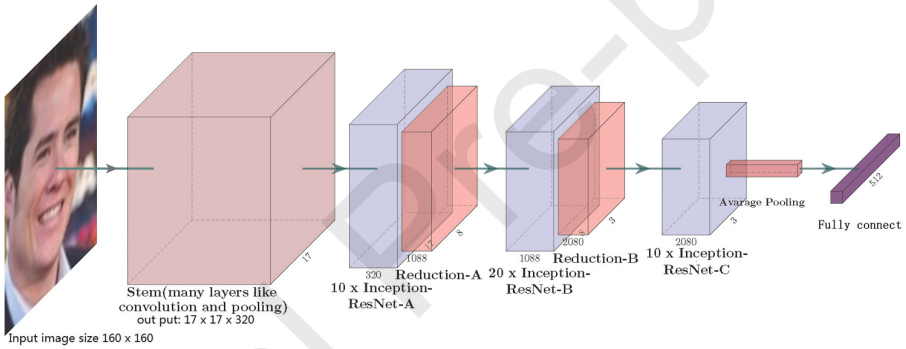
Face recognition is one of the most widely used applications in the field of computer vision. With the rapid development of artificial intelligence, there are more and more face authentication and recognition applications in the fields such as online payment, security check, access control and forensic sciences etc. Deep neural network played a critical role in up-to-date biometric advancements. In recent decades, we have witnessed the flourish of convolutional neural network architectures. Since the success of AlexNet[1] in the ImageNet competition, various CNN models have been proposed to solve different computer vision problems, such as object detection[2, 3, 4, 5], object classification[1, 6, 7, 8, 9, 10, 11], semantic segmentation[12, 13, 14, 15, 16], and human pose estimation[17, 18, 19, 20, 21, 22], etc. With the emergence of more and more powerful hardware devices, large and high-quality datasets appeared, supporting more complex CNN models. The constantly proposed network models have greatly promoted the development of computer vision. CNN now has become the primary method for solving the problems in these fields. However, how to train stable networks and improve the performance of these models are still great challenges.

In this paper, we investigated the architecture of the Inception-ResNet[9] proposed by Google. The Inception-ResNet module is a combination of the Inception block and the ResNet[8] structure. The architecture is shown in Figure 1. ResNet module primitively introduced residual connections that make it possible to train deeper neural networks. The Inception block can get more information from varying scales of input images and ResNet can gain accuracy from considerably increased depth. The combination of Inception block and ResNet module greatly improved the performance of the architecture. In original Inception-ResNet module, Inception block is scaled by a manually set factor, the so-called residual scaling factor. However, the network exhibits instability during training and always "died" early. The authors hinted that using residual scaling factors between 0.1 and 0.3 seems to stabilize the training process. But the factor values were manually assigned before training and need to be carefully

tuned. Furthermore, model training still encounters the risk of collapse.




(a) architecture of Inception-ResNet v1




(b) architecture of Inception-ResNet v2

Figure 1: Architectures of Inception-ResNet v1 and v2. The architecture includes a stem block, several Inception-ResNet blocks, reduction modules and fully connected layer. The stem block of v1 and v2 has 7 layers and 9 layers of convolutional and maxpooling, respectively. Inception-ResNet A, B and C are modularized blocks, which comprised of a series of convolutional and maxpooling layers.

We believe that the residual scaling factor can be set to a trainable parameter such that the factor can be learnt automatically rather than manually assigned. Adding only few trainable parameters will not noticeably increase the computational complexity of the model. On the contrary, it is helpful to improve the accuracy of the network and lead to better generalization.

The activation functions used in the Inception-ResNet module are ReLU

functions. ReLU is simple and very effective in most cases, however, arbitrarily forcing all outputs of negative inputs to be zero will make many neurons "dead" during training, thus damage the capability of the neural net. Leaky ReLU[23] was proposed to solve this problem and has been widely used in CNN. Leaky ReLU introduced a very small slope for negative part of the activation thus allowing gradient based update for negative input. Leaky ReLU can alleviate the problem of dead neurons. But the value of the negative part slope was set manually as a hyperparameter. In this paper, we also try to learn this value automatically during training following the approach of PReLU(Parametric Rectified Linear Unit)[24]. The basic motivation is that we try to decrease the impact of manually set parameters as much as possible. Therefore we propose our model of nearly fully trainable Inception-ResNet. To further promote the discriminative power of our model, we following ArcFace[25] to adopt Additive Angular Margin Loss as the loss function. We conducte comprehensive experiments on some popular large-scale image datasets, e.g. LFW[26, 27] and IJB-B[6]4. The results show that the proposed more trainable Inception-ResNet module can improve the performance of the network. Comparing the results of experiments on different versions of the network illustrates that the performance of the models using automatically learned parameters outperform the original Inception-ResNet models. It is also observed that the convergence speed is a little bit faster.

The major contributions of this work are three-fold: 1) We improve the Inception-ResNet model by setting the residual scaling factor to a trainable parameter. Initializing the factor with a small value and iteratively increasing the value slowly can avoid unstable training and relief the limitation imposed to the residual blocks. 2) We alternate the ReLU activation function with PReLU to make better use of the input information. As observed in experiments, the values converged to either positive values or negative values, which means the proposed method can utilize negative correlations more efficiently and reduce possible redundancy of the convolution kernel. 3) We introduce more trainable parameters to leverage the performance of the network. Despite slightly

increased the number of parameters, a model of more trainable parameters can improve the training stability and performance significantly. This also hints that exploring the possibility of training more hyperparameters such as architecture of models and the rate of optimizers is valuable.

The rest of the paper is organized as follows. Section II briefly reviews the history of the CNN framework architectures and recent advances in CNN-based face recognition. Section III describes the details of our proposed approach. Section IV presents the experimental results and analysis. Section V concludes our paper.

## 2. Related work

Since AlexNet[1] won the champion in ImageNet 2012 image classification challenge, convolutional networks have becoming increasingly popular in computer vision. Various frameworks of convolutional neural networks have been proposed. NIN[28] first introduced a 1x1 convolution which can significantly decrease the number of network parameters and thus improve the computational efficiency. Moreover, adding an activation function after each 1x1 convolutional layer can increase the nonlinearity of the network. VGGNet[6] performed a thorough evaluation of networks by increasing the depth using small (3x3) convolution filters and concluded that stacking more convolutional layers(16-19 layers) can achieve significant improvement. They also demonstrated that large convolution kernels could be decomposed by several small kernels. This decomposition can reduce the amount of network parameters while maintaining similar or even better performance. GoogLeNet[7] introduced an Inception module, which modularizes network structures and facilitates design of network architectures. ResNet[8] proposed an idea of modeling residuals, which alleviates the vanishing gradient problem and makes it possible to train very deep networks. Subsequently proposed Inception-ResNet[9] combined the Inception modules and the idea of residual connection to deepen and widen networks and achieved outperforming results. DenseNet[10] went even further than ResNet to

propose dense connections among network layers. A variety of works show that shortcut connections can reduce the number of parameters, speed up training process, and mitigate the problem of overfitting.SENet[11] introduced attention mechanism into CNN. They proposed Squeeze-and-Excitation blocks to weight defferents channals of the features. By this means, the global information can be used to selectively emphasize informative features and suppress less useful ones.

Deep convolutional neural networks greatly promoted the development of face recognition. Recent researches go deep into some critical problems encountered in face recognition with tremendous efforts. Li et al[29] proposed a two-stage approach consisting of data cleaning and multi-view deep representation learning. The data cleaning method can effectively reduce the noise level of the training data and thus improve the performance of deep-learning-based face recognition models. The multi-view representation learning enables the learned face representations to be more specific and discriminative. To treat multiple tasks jointly, Li et al[30] proposed an integrated Face Analytics Network (iFAN) to boost the performance by facilitate the informative interaction among different tasks. They also introduced a cross-dataset hybrid training strategy to solve the problem of absence of datasets for varying tasks, which allows tasks to use multiple datasets annotated for different purposes without requiring a fully labeled common dataset model. Enforced Softmax[31] is able to learn discriminative yet generative compact vector representations for face recognition, and boost the low-shot learning face recognition performance in presence of large multi-modality variances. Some work try to utilize synthetic human faces to solve the problem of lacking labeled samples, especially profile faces. DA-GAN[32] uses a Dual-Agent Generative Adversarial Network model to output realistic face simulators using unlabeled real faces while preserving the identity information. Synthesizing realistic profile faces is considered to be promising for pose-invariant models and unconstrained face recognition. Needless to say, loss function is of great importance in the training of a classification task. Many novel loss functions are proposed in face recogniting. [32] put for-

ward a couple of losses including pose perception loss, identity perception loss and adversarial loss, all of them can improve the performance for face recognition in particular scenaioes. ArcFace[25] uses Additive Angular Margin Loss to obtain highly discriminative features and achieved impressive results on many popular benchmarks.

Inception-ResNet module achieved remarkable performance in image classification and object detection. From a certain point of view, every Inception-ResNet module can be seen as a small CNN. The network architecture of Inception-ResNet is mainly composed of these small CNNs and some other network layers like pooling and convolution. Intuitively, if the Inception-ResNet module is improved, the entire network architecture can be improved. The structure of the Inception-ResNet module can be divided into two parts: the shortcut connection and the residual block. The shortcut connection directly maps the input features to the output and the residual block approximates the residual function. The final output of the module is the identical map of the input features plus the output of the residual block. [9] found in their experiments that if the number of layers exceed some specific value, the training stability of the network deteriorates rapidly. Unfortunately, this instability is difficult to handle by using lower learning rate or adding batch-normalization. They finally struggled to find an approach by scaling down the residuals before adding them to the final outputs. However, they picked the value of the scaling factors arbitrarily between 0.1 and 0.3. Obviously, the residual scaling factors benefit the training stability. But leaving it as a hyperparameter is a little bit hasty and may limit the performance of the network. In this paper, we consider the residual scaling factor as a trainable parameter and learn its value with the training process. Experimental results show that this can improved the accuracy.

Among all the components of network architectures, activation functions are crucial parts that have remarkable influence on network performance. ReLU functions are very popular in CNN for their simplicity and efficiency. The Inception-ResNet module also uses ReLU as acitivation function. ReLU forces all outputs of negative inputs to zero and thus the gradients of negative inputs

becoming zero accordingly, this will inevitably lose data information and damage training to a certain extent. Leaky ReLU[23] was first proposed in 2013 in an acoustic model and has been used in CNN. A leaky ReLU layer performs a threshold operation, where any input value less than zero is multiplied by a fixed scalar $\alpha$. However, the scalar is a hyperparameter and need to be artificially set before training. To address this issue, PReLU[24] proposed by He et al. turned the scalar $\alpha$ into a trainable parameter, and obtained better performance. We tried to replace the ReLU function in Inception-ResNet module with PReLU and trained a new version of the model. The idea is consistent with the method of training the aforementioned residual scaling factor, i.e., try to learning parameters as possible as we can.

## 3. Methodology

### *3.1. Trainable residual scaling factor (TRSF)*

We believe that if the parameter of the residual scaling factor is set to be trainable, the latent information that lies in the training set could be used to train a more stable network. As mentioned in [9], the value of residual scaling factor must not be too large, otherwise the training may be divergent. They set the value between 0.1 and 0.3 and all the modules share a same value for all residual scaling factors. We initialize the value of the factor to be 0.1 and update it with the training process. The small value of the factor ensures the stability at the early stage of training. With the training goes on, the factor will automatically adjust to an optimal value. Furthermore, the factor of each module can learn a specific value, which distinguishes a lot from the case of sharing value in many modules. Each finally optimized value of the residual scaling factor is applied to its corresponding Inception-ResNet module, which can make better use of potential capability of each module and improve the performance of the entire model. The improved Inception-ResNet module is shown in Figure 2.

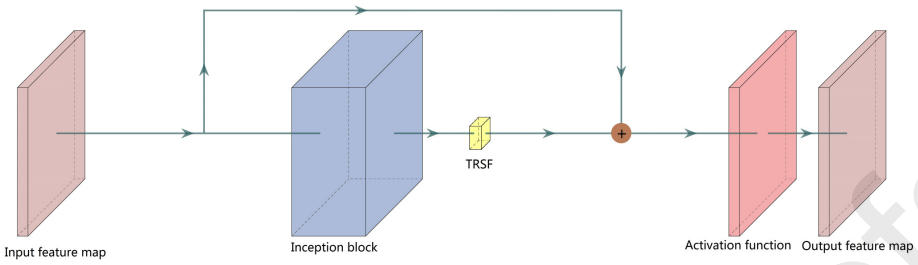



Figure 2: The structure of Inception-ResNet module with TRSF. The module has two branches, a direct map and a residual block. The residual block is scaled by TRSF and then summarized with the direct map.

During forward propagation of the network, following formula can be used to represent the value of the residual branch scaled by the trainable residual scaling factor:

$$y = TRSF \cdot Inception(x), \tag{1}$$

Where $TRSF$ represents the trainable residual scaling factor, $x$ and $Inception(x)$ represents the input and output of the residual inception block, respectively, and $y$ is the scaled output.

During back propagation of the network, it is easy to know that the gradient of $y$ with respect to $TRSF$ equals to $Inception(x)$, as Equation 2. And the gradient of the final object function, namely, $J$, can be calculated by Equation 3. Thus we can update $TRSF$ by iteration of gradient descent, shown in Equation 4.

$$\frac{\partial y}{\partial TRSF} = Inception(x), \tag{2}$$

$$TRSF \leftarrow TRSF - \eta \cdot \frac{\partial J}{\partial TRSF}, \tag{3}$$

$$\frac{\partial J}{\partial TRSF} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial TRSF}, \tag{4}$$

Where $\eta$ is the learning rate, $J$ is the loss of the entire model, $\leftarrow$ means update of the $TRSF$.

The total parameters of Inception-ResNet v1 and v2 are 22 million and 55 million, respectively. The number of TRSFs in Inception-ResNet v1 and v2 is 20 and 40, respectively. In this way, Adding the training parameters of TRSF in the network only increase 20 and 40 parameters, respectively, which is about one millionth of the total parameters. This shows that adding the trainable parameters of the TRSF just increment the total parameter by a very limited amount and scarcely increase the computational cost.

### *3.2. Module with Leaky ReLU and PReLU*

As mentioned earlier, activation functions have remarkable influences on model performance. Although ReLU function is widely used in convolutional neural network, it has some drawbacks, such as losing information and causing "dead" neurons. We intend to improve the activation function of the module by choosing Leaky ReLU and its variant PReLU instead of ReLU. The output of negative axis of ReLU is arbitrarily set to 0, which causes the ReLU neurons that have negative input incapable of updating during training and die directly. Both Leaky ReLU and PReLU substitute the negative axis by a small slop, which obviously reduces the loss of feature information and mitigates the problem of zero gradients.

The leaky ReLU function is shown as below:

$$y_i = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ \alpha_i x_i, & \text{if } x_i < 0 \end{cases}, \tag{5}$$

Where $\alpha_i$ is a fixed number. In our experiments the value is set to 0.2.

The formula for PReLU is similar to Equation 5 , except $\alpha_i$ in PReLU is a trainable parameter. Compared to ReLU, in spite of adding one extra parameter may increase the computational cost for a little bit, Leaky ReLU and PReLU can overcome the drawbacks of ReLU and improve the model performance apparently. Rather than set the value of $\alpha_i$ manually, we allow the parameter to be trainable too. Thus the value could be updated automatically during the training iteration.

## 4. Experiments

### *4.1. Datasets*

The training datasets we used is VGGFace2[33] and MS1MV2[25]. The VGGFace2 dataset includes 3.3 million face images from 9,131 individual person, with an average of 362 images for each subject. The images cover a wide range of ethnicities, accents, professions and ages, and the overall data is very noisy. The MS1MV2 dataset is a semi-automatic refined version of the MS-Celeb-1M[34] dataset. This dataset includes 5.8 million face images from 85,742 individual person and with higher data quality compared to VGGFace2.

The test datasets we used are the LFW[26, 27] and IJB-B[35] dataset. The LFW dataset contains more than 13,000 unconstrained face images and the IJB-B dataset contains 1,845 subjects with 21.8K still images and 55K frames from 7,011 videos. Both datasets are collected from the web, including varying pose, expression, lighting, age, gender and occlusion. The images of these datasets can be viewed as very similar to everyday lives, thus is very suitable for evaluation of all kinds of face verification and recognition algorithms. The test dataset we used is the LFW[26, 27] dataset. The LFW dataset is a face image database with a total of more than 13,000 unconstrained face images. The images were collected from the web, including varying pose, expression, lighting, age, gender and occlusion. The images of the LFW dataset can be viewed as very similar to everyday lives, thus is very suitable for evaluation of all kinds of face verification and recognition algorithms.

The VGGFace2 and MS1MV2 training data are first processed using MTCNN[36], and the faces in each picture are cropped to just fit the face objects. In order to ensure the fairness of the test, the face images of the test datasets are also processed in the same way.

### *4.2. Network architecture selection and setting*

The Inception-ResNet network proposed in [9] has two versions, v1 and v2. The Inception-ResNet modules used in the v1 and v2 versions are different. To

demonstrate the performances in more detail, we conducted experiments on the two versions of the network architectures, respectively. The v1 and v2 versions contain 20 and 40 Inception-ResNet modules, respectively. Most parameters of the network lie in these Inception-ResNet modules. Therefore, modifying the residual scaling factor and the activation function of the modules will not change the network architecture.

We conducted experiments on VGGFace2 training dataset with six different settings of the model. The setting are as follows: the original Inception-ResNet, Inception-ResNet with TRSF, Inception-ResNet with Leaky ReLU, Inception-ResNet with PReLU, Inception-ResNet with TRSF and Leaky ReLU, and Inception-ResNet with TRSF and PReLU. We also experimented on the MS1MV2 dataset with two different settings: Inception-ResNet with PReLU and Inception-ResNet with PReLU and TRSF.

We first used the traditional softmax-loss as the loss function and moved to ArcFace-loss to improve the performance. The input image size of the networks on VGGFace2 dataset is set to 160×160×3, and on MS1MV2 dataset is set to 112×112×3. This design can reduce the cost of the network and speed up the network model. The output is adjusted to a 512-dimensional feature, which contains plenty of information thus capable of dealing with large number of objects. Need to be mentioned, when training on the MS1MV2 dataset we removed a down-sampling layer from the stem block compared with that of training on the VGGFace2 dataset.

### *4.3. Training*

We trained our model on a workstation with two NVDIA TITAN RTX GPU using TensorFlow[37]. The optimizer we used is the mini-batch stochastic gradient descent with momentum. To regularize the parameters, we used weight decay of 0.9. On VGGFace2, The learning rate was initialized to 0.05 and decayed every 20 epochs using an exponential rate of 0.2. The batch size was set to 85, and we totally trained the network for 100 epochs, each epoch has 1000 iterations. For data augmentation, the training images are randomly flipped and

rotated before being sent to the network. We performed model evaluations on LFW testing set after every training epoch. We used the tool of Tensorboard[37] to visualize and monitor the training process. On MS1MV2, the learning rate starts from 0.1 and is divided by 10 at 50K, 80K, 110K iterations. The training process is finished at 170K iterations.

### *4.4. Experimental results and analysis*

#### *4.4.1. Evaluation results*

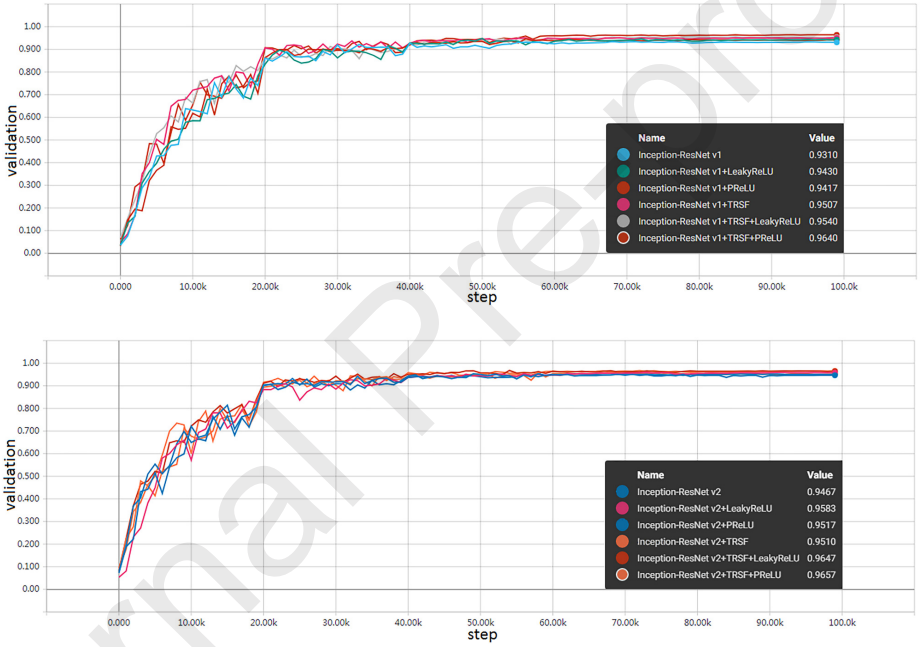



Figure 3: Validation of 6 different settings of softmax-loss-based Inception-ResNet v1 and v2 trained on VGGFace2 dataset. The "Value" refers to the face verification TAR(@FAR=1e-3) on LFW dataset. **Top**: Validation of 6 different settings of Inception-ResNet v1. **Bottom**: Validation of 6 different settings of Inception-ResNet v2.

To compare the performance of the proposed softmax-loss-based method on VGGFace2 training dataset, we conducted thorough experiments on the LFW dataset. We mainly evaluated twelve models by the face verification accuracy, i.e., the original Inception-ResNet v1 and v2, and their improved version by us-

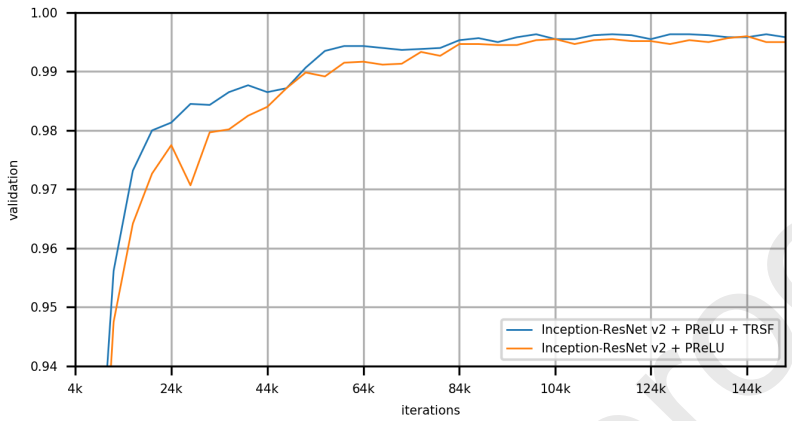



Figure 4: Validation of 2 different settings of ArcFace-loss-based Inception-ResNet v2 trained on MS1MV2 dataset. The value of validation refers to the face verification TAR(@FAR=1e-3) on LFW dataset.

ing LeakyReLU, PReLU, TRSF, TRSF+LeakyReLU, TRSF+PReLU, respectively. The experiments on the same dataset were conducted under same conditions. The verification accuracy curves during training are show in Figure 3. The reported face verification TAR (@FAR=1e-3). The TRSF was initialized to 0.1 and the coefficients of the negative parts of LeakyReLU and PReLU were initialized to 0.2. Table 1 and Table 2 show the face verification TAR (@FAR=1e-3) of the final models testing on LFW test dataset. As we can see, the accuracy improves step-by-step when using trainable parameters. The verification accuracy of the Inception-ResNet v1 + TRSF + PReLU and the Inception-ResNet v2 + PReLU + TRSF achieved to 95.90% and 96.53% TAR (@FAR=1e-3), respectively. It is worth noting that when using trainable parameters of the TRSF and the coefficient of the negative part of PReLU there is no crash situation during training which may be encountered in the original model.

We also trained a model based on Additive Angular Margin Loss(ArcFace-loss)[25] on the MS1MV2 dataset and evaluated on the IJB-B dataset. As show in table 3, the TAR(@FAR=1e-4) of Inception-ResNet v2 + PReLU + TRSF

increased from 90.65% to 91.49% compared with Inception-ResNet v2 + PReLU, indicating that the introduce of the factor TRSF provide more generalization on different datasets. To compare the training process of these two methods more clearly, we illustrated the convergence of the validation on the LFW dataset, as shown in figure 4.

Table 1: Verification TAR of different settings of softmax-loss-based Inception-ResNet v1 trained on VGGFace2 dataset.

| Network model | TAR(@FAR=1e-3) |
|---|---|
| Inception-ResNet v1 | 93.40% |
| Inception-ResNet v1 + TRSF | 95.23% |
| Inception-ResNet v1 + Leaky ReLU | 94.30% |
| Inception-ResNet v1 + TRSF + Leaky ReLU | 95.50% |
| Inception-ResNet v1 + PReLU | 94.53% |
| Inception-ResNet v1 + TRSF + PReLU | 95.90% |

Table 2: Verification TAR of different settings of softmax-loss-based Inception-ResNet v2 trained on VGGFace2 dataset.

| Network model | TAR(@FAR=1e-3) |
|---|---|
| Inception-ResNet v2 | 95.33% |
| Inception-ResNet v2 + TRSF | 95.97% |
| Inception-ResNet v2 + Leaky ReLU | 95.67% |
| Inception-ResNet v2 + TRSF + Leaky ReLU | 96.43% |
| Inception-ResNet v2 + PReLU | 95.83% |
| Inception-ResNet v2 + TRSF + PReLU | 96.53% |

Although manually set parameter value may introduce some prior to the model, automatically learn by data can improve the model precision. Moreover, too many manually set parameters may further damage the interpretability of neural networks. Data driven automatic learning of some hyper parameters may help to improve the performance of the model. This assertion can be illustrated

Table 3: Verification TAR of different settings of ArcFace-loss-based Inception-ResNet v2 trained on MS1MV2 dataset.

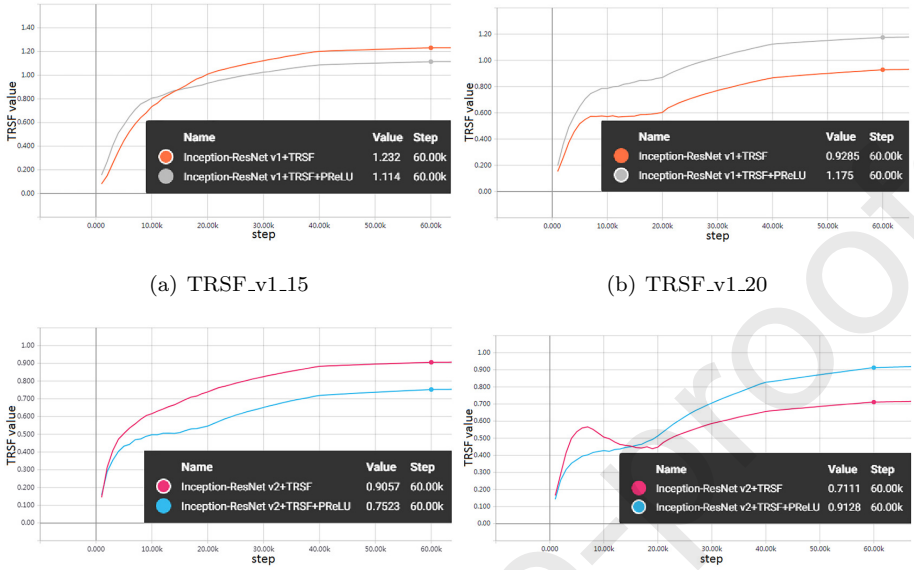
| Network model | TAR(@FAR=1e-4) |
|---|---|
| Inception-ResNet v2 + PReLU | 90.65% |
| Inception-ResNet v2 + TRSF + PReLU | 91.49% |

by. From Table 1 and Table 2 we can see that the automatic learning of TRSF has a 1.24% promotion. PReLU and Leaky ReLU have 0.82% and 0.62% improvement to the network model, respectively. Jointly using the learnable TRSF and the negative part of PReLU can further improve the model accuracy. This also indicates that the values of PReLU and TRSF are not conflicted and more learnable parameters can leverage the model performance.

*4.4.2. TRSF*

We set the TRSF parameter to be learnable in training. Following the initial values of the residual scaling factor in [9], we initialize all the values of the TRSF to 0.1. All the values of TRSF converged during training (See Figure 5), and most values iteratively approximated between 0.6 and 1.3, as shown in Figure 6 and Figure 7. Figure 6 and Figure 7 show all the TRSF values after 100 epoch training in the v1 and v2 versions, respectively. We calculated the average values of the TRSF in v1 and v2 with different activation functions. The results are shown in Table 4 and Table 5. As we can see, the values of the TRSF converged to a positive value around 1.0. This means that even given a small initial value of 0.1, the parameters increased iteratively. The small initial value can ensure the training to be stable. TRSF iteratively converges to big positive value means that the branches of residual block can eventually play major roles in the model. Although the TRSF converged to different values in different versions of the proposed methods, all the factors were updated to relatively bigger values. Compare to the original model of [9], the importance of the residual blocks have increased by an order of magnitude.

The motivation of [9] setting a relatively small value of the residual scaling




(a) TRSF_v1_15 (b) TRSF_v1_20

(c) TRSF_v2_30 (d) TRSF_v2_39

Figure 5: Visualization of the training process for TRSF values in different modules of the softmax-loss-based Inception-ResNet model on VGGFace2. TRSF is named as "TRSF_inception-renet versionID_blockID".

Table 4: Average TRSF values of different modules in Inception-ResNet v1.

| Network model | Average value of TRSF |
|---|---|
| v1 TRSF | 1.014 |
| v1 TRSF + Leaky ReLU | 0.954 |
| v1 TRSF + PReLU | 1.029 |

Table 5: Average TRSF values of different modules in Inception-ResNet v2.

| Network model | Average value of TRSF |
|---|---|
| v2 TRSF | 0.772 |
| v2 TRSF + Leaky ReLU | 0.765 |
| v2 TRSF + PReLU | 0.738 |

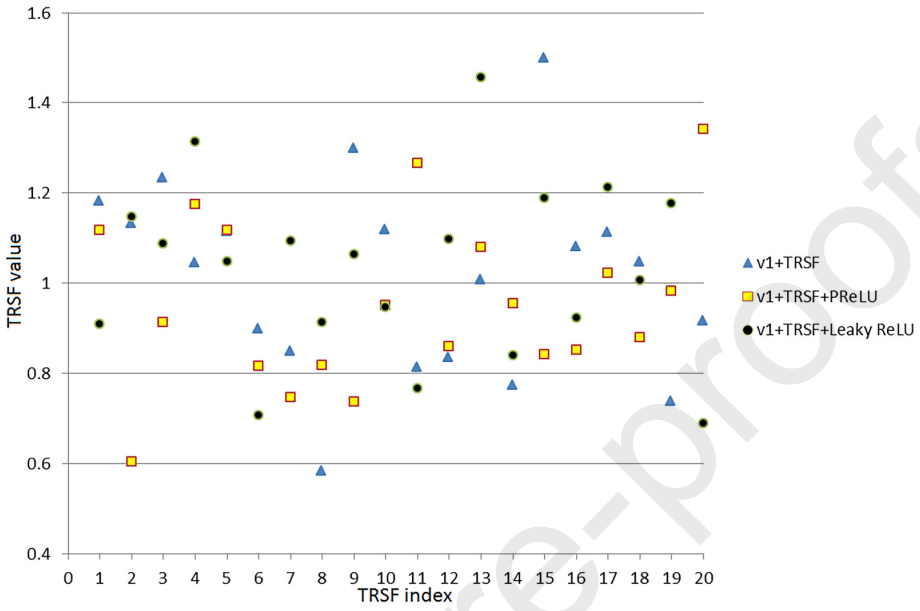



Figure 6: TRSF values of different modules in Inception-ResNet v1.

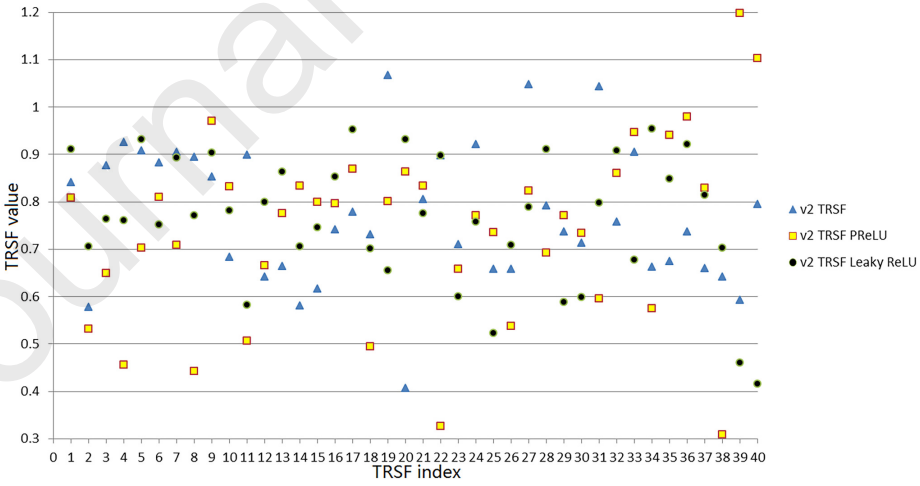



Figure 7: TRSF values of different modules in Inception-ResNet v2.

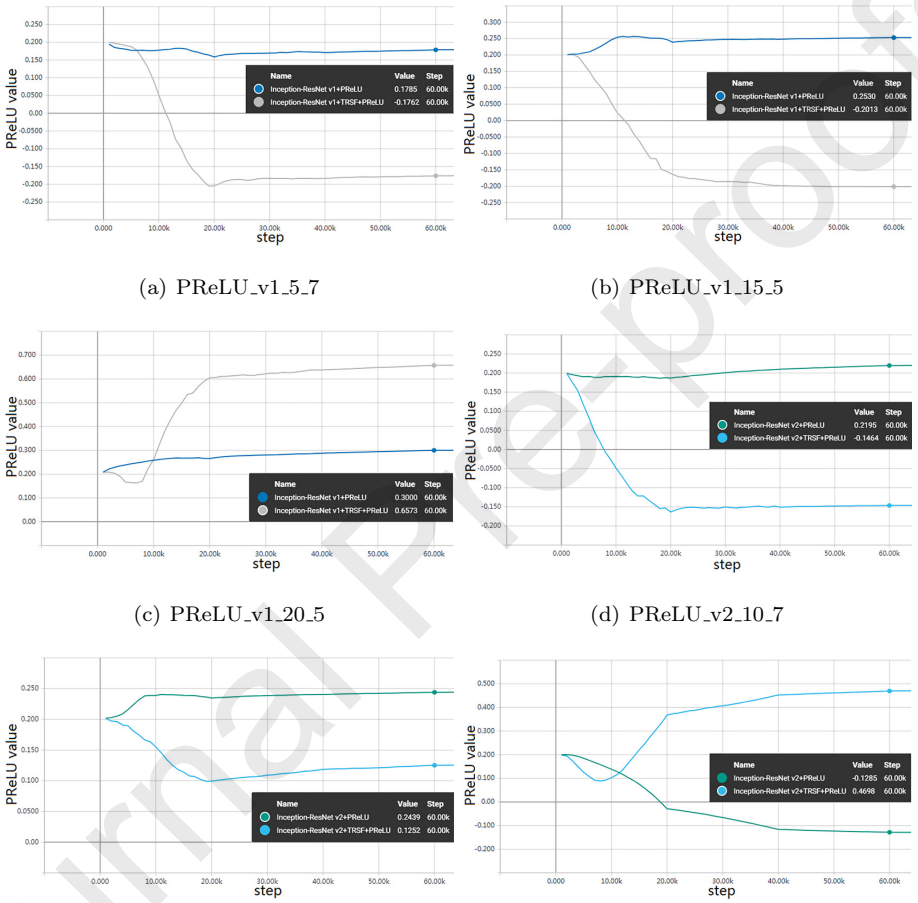
factor is to ensure the stable training process and avoid crash in early training. But the small values of the residual scaling factor inevitably weaken the importance of the residual blocks which have main numbers of parameters and should take more active role in the model. The proposed method similarly set small values at the beginning and iteratively trained to increase the values slowly. Thus can avoid unstable training and relief the limitation imposed to the residual blocks.

#### *4.4.3. PReLU*

Activation functions are essential non-linear mapping for extracting features. Traditionally, the activation function uses ReLU or sigmoid functions. But these activations may cause many issues such as gradient vanishing or dead neurons. Leaky ReLU can alleviate this problem, however, the slope of the negative part is manually assigned. We use PReLU as the activation function and thus all the parameter of the activation are automatically learned from training data.

We initialized the value of the negative part of PReLU to 0.2 and iteratively update the value with the training process. Figure 8 visualized the update process of some trained values. As we can see, the slope value of the negative part converges eventually. It is worth noting that some values converged to negative values. As early research work[38, 39, 40] shows, convolutional layers learn both positive and negative phase information of the input. The results of our experiments show that some slopes of negative part of PReLU practically converged to negative values (See figure 8,9,10). This means that by automatically learned PReLU the negative phase information actually play a role. Compared to ReLU, both Leaky ReLU and PReLU utilize negative phase information to leverage the ability of the model and avoid the problem of dead neurons. Furthermore, the learned values of PReLU can make good use of the negative correlations and reduce the possible redundancy of the convolution kernel.

To test the sensibility of learnable PReLU to initial values, we trained 5 models of Inception-ResNet v2 + PReLU with different initial values. The initial values -0.1, 0.1, 0.2, 0.25, 0.3 were used. Table 6 gives the performance of




(a) PReLU_v1_5_7 (b) PReLU_v1_15_5

(c) PReLU_v1_20_5 (d) PReLU_v2_10_7

(e) PReLU_v2_30_5 (f) PReLU_v2_39_5

Figure 8: Training curves of same PReLU $\alpha$ values in different modules of different settings of softmax-loss-based Inception-ResNet model on VGGFace2. The subtitles are named as "PReLU_inception-renet versionID_blockID_PReLUID"

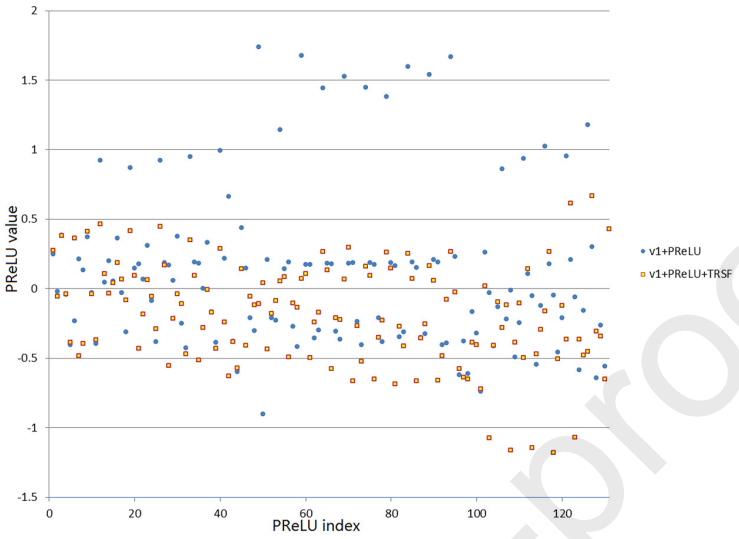



Figure 9: $\alpha$ values of PReLU in Inception-ResNet v1 + PReLU and Inception-ResNet v1 + PReLU + TRSF. Each model consists 131 $\alpha$ values of PReLU. The $\alpha$ values of v1+PReLU includes 69 positive values and 62 negative values. The $\alpha$ values of v1 + PReLU + TRSF includes 42 positive values and 89 negative values.

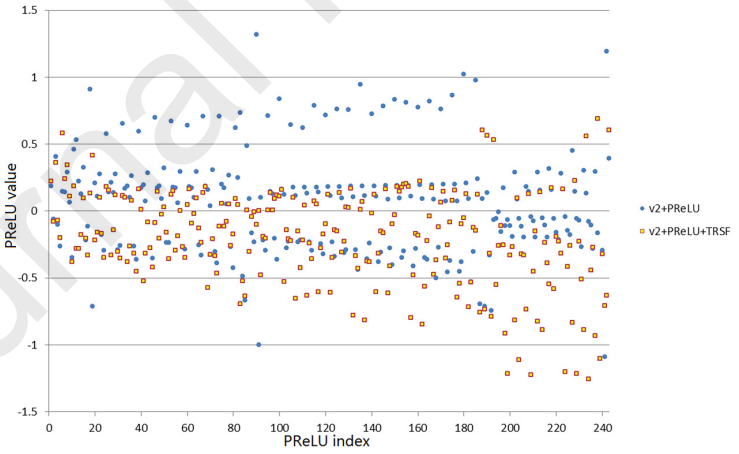



Figure 10: $\alpha$ values of PReLU in Inception-ResNet v2 + PReLU and Inception-ResNet v2 + PReLU + TRSF. Each model consists 243 $\alpha$ values of PReLU. The $\alpha$ values of v2 + PReLU includes 138 positive values and 105 negative values. The $\alpha$ values of v2 + PReLU + TRSF includes 81 positive values and 162 negative values.

Table 6: Verification TAR of different initial PReLU a values of Inception-ResNet v2 + ReLU. The evaluation is conducted on the LFW at TAR(@FAR=1e-3).

| Initial value | TAR(@FAR=1e-3) |
|---|---|
| -0.1 | 94.57% |
| 0.1 | 94.37% |
| 0.2 | 95.83% |
| 0.25 | 95.97% |
| 0.3 | 94.57% |

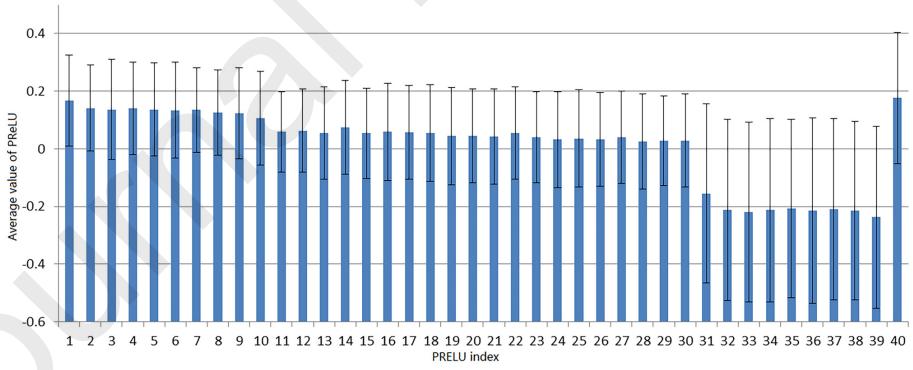



Figure 11: Five Inception-ResNet v2+PReLU models with different initial PReLU $\alpha$ values, we selected 40 PReLU $\alpha$ value out of 243 to visualize the histogram of the mean and standard deviation.

the models with different initial values. Figure 11 illustrates the average value and standard variation of the negative part of PReLU with different models. The results show that the initial value of 0.25 has the best performance, which is same as the best initial value proposed in [24]. There is a difference of 1.6% in the validation rate for the best and the worst model. This result indicates the model is sensitive to the initial value to some extent.

## 5. Conclusion

The first idea presented in this paper is to change the residual scaling factor in Inception-ResNet from a hyperparameter to a trainable parameter. A residual scaling factor with a small initial value (such as 0.1) ensures the stability of the training network at the beginning. The residual scaling factor for each Inception-ResNet module will be trained to a value appropriate for its own module. Experiments show that this change can improve the performance of the network. The second idea is using Leaky ReLU and PReLU in the Inception-ResNet module. It is found that Leaky ReLU and PReLU can make better use of input information and improve network performance.

While observing the problem of instability in training Inception-ResNet, we improved the model by using more trainable parameters. Both the residual scaling factor and the negative part of ReLU activation are set to be learnable to avoid the crash situation during training the model. We initialize the parameters with small values and update with the iteration of training. Extensive experiments suggest that the proposed method can ensure stability of the model and improve performance simultaneously. Furthermore, most of the trained residual scaling factors converge to relatively large values. This indicates that the residual block can play a more import role in the model. We argue this is the main reason that the proposed model has better performance. Some of the learned values of PReLU are negative. The result hints that the trainable PReLU can make good use of the negative correlations and reduce the possible redundancy of the convolution kernel. Future work may explore the possibility

of training more hyperparameters such as the architecture of the model and the learning rate of optimizer, etc.

**Acknowledgments**

This work is supported by Beijing municipal education committee scientific and technological planning project (KM201811232024, KM201611232022) and Beijing excellent talents youth backbone project (9111524401). We would also like to thank Qinglin Ran, Zhiying Hu and Mingyang Yuan for their support and helpful comments.

**References**

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105. doi:10.1145/3065386.

[2] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE transactions on pattern analysis and machine intelligence 37 (9) (2015) 1904–1916. doi:10.1109/TPAMI.2015.2389824.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, Springer, 2016, pp. 21–37. doi:10.1007/978-3-319-46448-0_2.

[4] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969. doi:10.1109/ICCV.2017.322.

[5] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

[6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[10] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708. doi:10.1109/CVPR.2017.243.

[11] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[12] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440. doi:10.1109/TPAMI.2016.2572683.

[13] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE transactions on pattern analysis and machine intelligence 39 (12) (2017) 2481–2495. doi:10.1109/TPAMI.2016.2644615.

[14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE transactions on pattern analysis and

machine intelligence 40 (4) (2017) 834–848. doi:10.1109/TPAMI.2017.2699184.

[15] G. Lin, A. Milan, C. Shen, I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1925–1934. doi:10.1109/CVPR.2017.549.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2881–2890. doi:10.1109/CVPR.2017.660.

[17] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4724–4732. doi:10.1109/CVPR.2016.511.

[18] A. Newell, K. Yang, J. Deng, Stacked hourglass networks for human pose estimation, in: European conference on computer vision, Springer, 2016, pp. 483–499. doi:10.1007/978-3-319-46484-8_29.

[19] Y. Chen, C. Shen, X.-S. Wei, L. Liu, J. Yang, Adversarial posenet: A structure-aware convolutional network for human pose estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1212–1221. doi:10.1109/ICCV.2017.137.

[20] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, B. Schiele, Deepercut: A deeper, stronger, and faster multi-person pose estimation model, in: European Conference on Computer Vision, Springer, 2016, pp. 34–50. doi:10.1007/978-3-319-46466-4_3.

[21] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, J. Sun, Cascaded pyramid network for multi-person pose estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7103–7112. doi:10.1109/CVPR.2018.00742.

[22] H.-S. Fang, S. Xie, Y.-W. Tai, C. Lu, Rmpe: Regional multi-person pose estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2334–2343. doi:10.1109/ICCV.2017.256.

[23] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, Vol. 30, 2013, p. 3.

[24] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123.

[25] J. Deng, J. Guo, N. Xue, S. Zafeiriou, Arcface: Additive angular margin loss for deep face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4690–4699.

[26] G. B. Huang, M. Mattar, T. Berg, E. Learned-Miller, Labeled faces in the wild: A database forstudying face recognition in unconstrained environments, 2008.

[27] G. B. Huang, E. Learned-Miller, Labeled faces in the wild: Updates and new reporting procedures, Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep (2014) 14–003.

[28] M. Lin, Q. Chen, S. Yan, Network in network, arXiv preprint arXiv:1312.4400 (2013).

[29] J. Li, J. Zhao, F. Zhao, H. Liu, J. Li, S. Shen, J. Feng, T. Sim, Robust face recognition with deep multi-view representation learning, in: Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 1068–1072.

[30] J. Li, S. Xiao, F. Zhao, J. Zhao, J. Li, J. Feng, S. Yan, T. Sim, Integrated face analytics networks through cross-dataset hybrid training, in: Proceedings of the 25th ACM international conference on Multimedia, 2017, pp. 1531–1539.

[31] Y. Cheng, J. Zhao, Z. Wang, Y. Xu, K. Jayashree, S. Shen, J. Feng, Know you at one glance: A compact vector representation for low-shot learning, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 1924–1932.

[32] J. Zhao, L. Xiong, P. K. Jayashree, J. Li, F. Zhao, Z. Wang, P. S. Pranata, P. S. Shen, S. Yan, J. Feng, Dual-agent gans for photorealistic and identity preserving profile face synthesis, in: Advances in neural information processing systems, 2017, pp. 66–76.

[33] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, A. Zisserman, Vggface2: A dataset for recognising faces across pose and age, in: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), IEEE, 2018, pp. 67–74. doi:10.1109/FG.2018.00020.

[34] Y. Guo, L. Zhang, Y. Hu, X. He, J. Gao, Ms-celeb-1m: A dataset and benchmark for large-scale face recognition, in: European conference on computer vision, Springer, 2016, pp. 87–102.

[35] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. K. Jain, J. A. Duncan, K. Allen, et al., Iarpa janus benchmark-b face dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 90–98.

[36] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Processing Letters 23 (10) (2016) 1499–1503. doi:10.1109/LSP.2016.2603342.

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467 (2016).

[38] W. Shang, K. Sohn, D. Almeida, H. Lee, Understanding and improving

convolutional neural networks via concatenated rectified linear units, in: international conference on machine learning, 2016, pp. 2217–2225.

[39] K. Jarrett, K. Kavukcuoglu, Y. LeCun, et al., What is the best multi-stage architecture for object recognition?, in: 2009 IEEE 12th international conference on computer vision, IEEE, 2009, pp. 2146–2153. doi:10.1109/ICCV.2009.5459469.

[40] S. K. Roy, S. Manna, S. R. Dubey, B. B. Chaudhuri, Lisht: Non-parametric linearly scaled hyperbolic tangent activation function for neural networks, arXiv preprint arXiv:1901.05894 (2019).