LinArc - Deep Face Recognition Using LinCos And ArcFace

Ravi Chopra, Joydip Dhar, Vinal Patel
Computer Science and Engineering
Atal Bihari Vajpayee Indian Institute of Information Technology and Management Gwalior,
Madhya Pradesh, India 474015
mailtoravi1002@gmail.com, jdhar@iiitm.ac.in, vp@iiitm.ac.in

***Abstract*—Data is overflowing day by day. The use of face recognition is rapidly picking up pace due to the boom in data and available computation power. The research done in this field is at an unimaginable pace, and accuracies of more than 99% have been achieved, which are possibly less only by Baye's error. However, there is still room for experimentation. This paper tries to build a model by mixing two novel ideas of face recognition - ArcFace and LinCos. In this paper, the target is to manipulate the Additive Angular Margin Loss used by ArcFace by incorporating the ideas of LinCos. We re-train the pre-trained ArcFace model using Mobile FaceNet with a modified loss function. The results suggest that our model optimizes at a faster rate as compared to the ArcFace and LinCos models.**

***Index Terms*—Face recognition, Loss functions, Face identification, ArcFace.**

I. Introduction

We are living in the age of data. This data can be used to ease the life of human beings. Face recognition using machines is one of the tasks which are possible due to immense growth in data and available computational power. Face verification has many applications ranging from biometric identification, emotion recognition, criminal identification, etc. Starting from eigenFaces [1] in 1991 to FaceNet [2] in 2015, research in this field is amazing. There are various methods proposed for face verification using Deep Convolutional Neural Network (DCNN). Broadly we can classify them into two approaches. One is that learn an embedding such as triplet loss [2] directly, and the other is to use a softmax classifier [3]–[5] to train a multi-class classifier that is able to distinguish between different identities in the dataset.

FaceNet is considered a significant milestone in the first approach, which uses siamese networks with triplet loss. However, due to the availability of a large amount of data and extensive computational power, the second approach, which proposes to train a classifier, has gained significant popularity. Metric learning loss functions require sample mining strategies, and hence the loss function used can heavily affect the classifier's performance. A good classifier is one that can group similar faces and separate dissimilar faces. Softmax is used as the most common loss function. However, the problem with the softmax loss function is that it only optimises inter-class variance but fails to minimise variation in different classes. To achieve the second objective, many ideas propose to use a margin penalty. This idea was pioneered by SphereFace [6] and further developed in ArcFace [7]. The main objective of ArcFace is to increase intra-class compactness and maximise inter-class dispersion. There are many approaches that tried to achieve the same before ArcFace. Some noteworthy mentions are Centreloss [8], SphereFace [6], and CosFace [9]. Centerloss proposed that the Euclidean distance between each feature vector and its class centre can be used to obtain intra-class compactness while the interclass dispersion can be achieved by the joint penalisation of the softmax loss. SphereFace was the first to introduce the idea of angular margin. It proposes to use a multiplicative angular margin penalty to enforce extra intra-class compactness and inter-class discrepancy simultaneously, which leads to a better discriminative power of the trained model. The idea is further explored in CosFace, where a cosine margin penalty is directly added to the target logit. CosFace achieved better performance than SphereFace with easier implementation. The idea was further explored in ArcFace that made the model engaging, effective, easy, and efficient. Moreover, the classifier in ArcFace is trained to distinguish between different identities using a modified version of softmax loss (Additive Angular Marginal Loss). The idea proposed in ArcFace suggests adding an additive angular margin to the target angle. The angle is obtained by taking arccosine of the dot product of the DCNN feature and the last fully connected layer. However, these models are prone to over-fitting due to the non-linear nature of cosine function. To tackle this, LinCos [10] was proposed, in LinCos linearization of the softmax loss function is conventionally used for training of classifiers. It makes use of Taylor's expansion to approximate cosine in softmax loss. This is done to counter the effect of the non-linear nature of cosine, which may hinder the network from thoroughly learning to maximise the angular discriminability of features.

In this paper, LinCos and ArcFace models are merged to develop the proposed LinArc as presented in Section II. Section III presents the experimental results, and the concluding remarks are made in Section IV.

II. Proposed Methodology

The proposed methodology presented in Fig. 1 is divided into three parts: Part 1, Part 2, and Part 3. In Fig. 1, x_i denotes the deep feature of the i-th sample for y_i-th class, and W_j denotes the j-th column of the weight W. In early face recognition systems, softmax is used a loss function for classification which is represented as follows:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}, \quad (1)$$

where N and n represent the batch size and the class number, respectively. However, it fails to give higher similarity for intra-class samples and diversify inter-class samples. To overcome this, both features and weights are normalised so that the predictions are dependent only on the angle between the feature and the weight as propopsed in [7]. First, the angle between the weight W_j and x_i is calculated, which is denoted by θ_j. $cos\theta_\mathrm{j}$ is obtained using the dot product of W_j and x_i as $W_j{}^T x_i$. Next, θ_j is obtained by taking arccosine of the dot product. This is equal to the angle between the feature x_i and the ground truth weight W_{y_i}. Recently, ArcFace proposed to train a classifier using a modified softmax loss function, given as [7]

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\left(\cos(\theta_{y_i}+m)\right)}}{e^{s\left(\cos(\theta_{y_i}+m)\right)} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}, \quad (2)$$

m is marginal penalty added between x_i (deep feature of the i-th sample belonging to the y_i-th class) and W_y (weight) which enhance the discriminative power of the trained classifier. This additive angular margin penalty is equal to the geodesic distance margin penalty in the normalised hypersphere. We have also included the margin penalty m is with the angle,θ_j and this completes part 1 of the proposed methodology. In the next stage, ArcFace proposes to take the cosine of the resultant angle obtained in part 1. However, this may have a negative impact on the overall performance of the classifier due to the non-linear nature of the cosine function. The cosine function being non-linear may lead to insufficient angular optimisation by over saturating the angle. This also hinders the network from learning discriminability between different classes and maximise intra-class compactness. A natural workaround is to use a linear logit in place of the cosine function. LinCos loss proposes a trade-off between linear logit and the cosine function. This trade-off is a direct result of replacing the cosine function by Taylor's expansion up to the first K terms. First, θ_j is calculated by taking arccosine of $cos\theta_\mathrm{j}$ as shown in (3). This angle is then approximated to first K terms of arccosine Taylor's expansion as shown in (4) and (5). The linear-cosine logit shown in (7) is derived by substituting θ_j with $\hat{\theta}_j$ in (6). Thus, the Linear-Cosine Softmax Loss (LinCos-Softmax) for the i-th sample is defined in (8) and (9).

$$\theta_j = \arccos\left(\cos\theta_j\right) \approx \hat{\theta}_j, \quad (3)$$

where

$$\hat{\theta}_j = \frac{\pi}{2} - \sum_{n=0}^{K-1} c_n\left(\cos\theta_j\right)^{2n+1}, \quad (4)$$

and

$$c_n = \frac{(2n)!}{2^{2n}(n!)^2(2n+1)}. \quad (5)$$

Now, θ_j in linear logit, defined in (6) is substituted by $\hat{\theta}_j$, defined in (4).

$$f_j^{\text{linear}} = -\theta_j + \frac{\pi}{2} = -\arccos\left(\cos\theta_j\right) + \frac{\pi}{2}, \quad (6)$$

$$f_j^{LinCos} = -\hat{\theta}_j + \frac{\pi}{2} = \sum_{n=0}^{K-1} c_n\left(\cos\theta_j\right)^{2n+1}, \quad (7)$$

$$L_i^{LinCos} = -\log\left(P_{y_i}^{LinCos}\right), \quad (8)$$

where

$$P_{yi}^{LinCos} = \frac{e^{s f_{yi}^{LinCos}}}{\sum_{j=1}^{C} e^{s f_j^{LinCos}}}. \quad (9)$$

As we can see, this helps reduce the negative impact of the non-linear nature of the cosine function. However, the use of Taylor's expansion up to K terms to approximate *cos(θ+m)* restricts our classifier to a fixed coefficient for each term. The loss function so obtained can be generalised to a further extent by replacing coefficients of each term in Taylor's expansion by $(-1)^n/\alpha_\mathrm{n}$ as shown in (11). The resultant value $\theta_{\mathrm{y_i}}$ is first approximated using (11) and then passed through a feature scale unit where all the logits are multiplied by the feature scale *s*. Finally, the scaled value is passed to a softmax unit and contributes to the cross-entropy loss. Therefore, by incorporating the idea of linearizing the $\cos(\theta_{y_i} + m)$ by Taylor series expansion in (10), the overall loss function can be written as

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\left(f\left(\theta_{y_i}+m\right)\right)}}{e^{s\left(f\left(\theta_{y_i}+m\right)\right)} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}, \quad (10)$$

where

$$f(\theta_{y_i} + m) = \sum_{n=0}^{K-1}\frac{(-1)^n(\theta_{y_i}+m)^{2n}}{\alpha_n} \quad (11)$$

A suggestive guide on how to choose values for K and α_n ($\forall n \in [0, K-1]$) is presented in the next section. This approximated value is feature-scaled and then passed to the softmax unit to contribute to cross-entropy loss as shown in Part-3 of the Fig. 1.

III. Experimental Results

The experiment has been done using the Labeled Faces in the Wild (LFW) benchmark [11], which is considered a standard in the field of face recognition. This dataset has more than 12,000 images in the training set and 1,000 images in the test set. In the first phase of the experiment, we train the pre-trained model by ArcFace without any modifications in the ArcFace head($m = 0.5$) on the LFW dataset. In the second phase, the ArcFace head is modified($m = 0.5$) as proposed in the previous section, and the pre-trained model by ArcFace is again trained on the LFW dataset. After that, the results are recorded.

Ideally, we must treat K and α_n as hyper-parameters and changed during training as per the validation loss. However, it is possible that there is a limitation of computational resources.

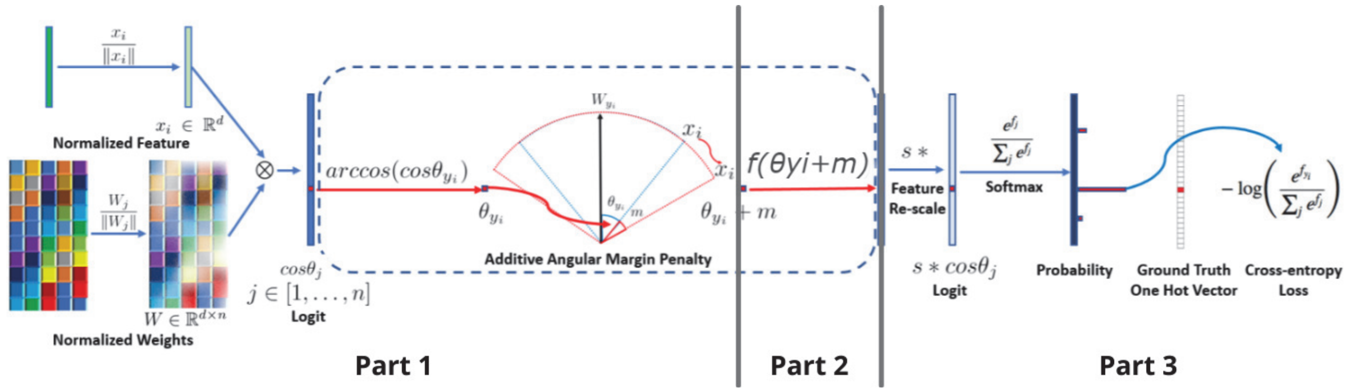

Fig. 1. Proposed LinArc pipeline, $f(\theta_{y_i} + m) = \sum_{n=0}^{K-1} \frac{(-1)^n(\theta_{y_i}+m)^{2n}}{\alpha_n}$

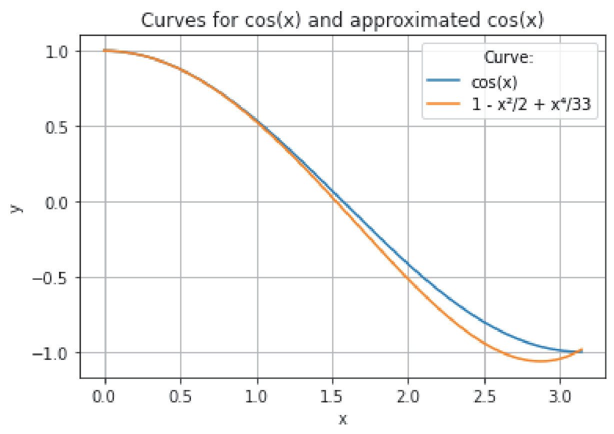

Fig. 2. Curves for $cos(x)$ and $1 - x^2/2 + x^4/33$

In that case, we can just replace $1/\alpha_n$ with $1/(2n)!$ (coefficients of the *n*-th term in Taylor's expansion) as suggested in LinCos.

The other way around is to choose a value of K and compare curves of $\sum_{n=0}^{K-1} \frac{(-1)^n(x)^{2n}}{\alpha_n}$ for various values of α_{n} ($\forall n \in [0, K-1]$) with the curve of $cos(x)$. The curve which gives the best approximation in the range of $0 <= x < \pi$ is chosen. This approach was used in this paper. After comparing various plots, $1 - x^2/2 + x^4/33$ was selected because it approximated $cos(x)$ sufficiently well, as shown in Fig. 2. Hence, we chose $K = 3$, $\alpha_0 = 1$, $\alpha_1 = 2$, and $\alpha_2 = 33$.

The experiment was conducted in two phases. In phase 1, the pre-trained model was trained on the LFW dataset with Mobile Face Net [12] as backbone without any modification in ArcFace head. An accuracy of 95.75% was achieved at the end of 20 epochs. In phase 2, the pre-trained model was trained in LFW dataset with Mobile Face Net as the backbone with proposed changes. An accuracy of 96.61% accuracy was achieved at the end of 20 epochs. The training was done using Tesla K80 GPUs provided by Amazon Web Service p2.xlarge instance. The results reported are using the same machine.

The increase in accuracy can be attributed to the introduced linearity in the proposed LinArc model. However, the exciting part is that the model with a modified pipeline converges faster as compared to the model with ArcFace head, as shown in Fig. 3. This can also be attributed to the linearity in the model.

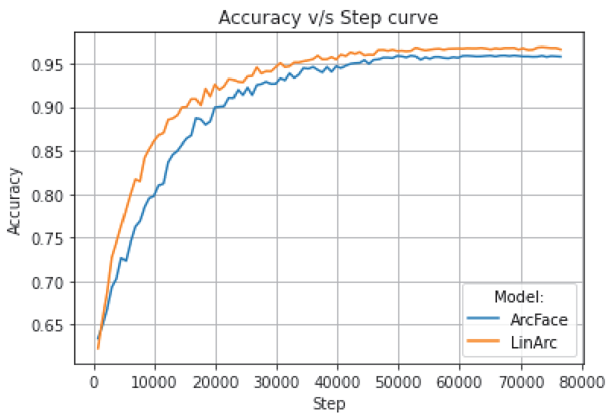

Fig. 3. Accuracy v/s Step curve

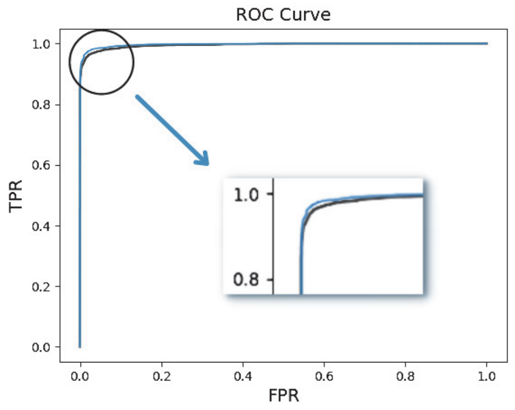

Fig. 4. ROCs curve for ArcFace (black) and Proposed LinArc (blue).

Finally, we plot Receiver Operating Characteristics (ROCs) for both the models in Fig. 4. From the zoomed version in Fig. 4, it can be seen that True Positive Rate (TPR) for the same False

Positive Rate (FPR) is higher for LinArc (Blue) as compared to ArcFace (Black).

IV. Conclusion

The work presented in this paper is a step forward to a new way of thinking in Face verification. The approach of the work is to merge two existing novel ideas presented by papers- ArcFace and LinCos. The results of the experiment suggest that replacing the cosine function in ArcFace with its lower-order Taylor's expansion may prove beneficial. We can achieve an accuracy almost equal to the state-of-art model in a smaller number of epochs.

References

[1] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 1991, pp. 586–587.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 815–823.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2014, pp. 1701–1708.

[4] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *Proceedings of the 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, 2018, pp. 67–74.

[5] O. M. Parkhi, A. Vedaldi, and A. Zisserman, *Deep face recognition*. British Machine Vision Association, 2015.

[6] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 212–220.

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

[8] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proceedings of the European conference on computer vision*, 2016, pp. 499–515.

[9] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "CosFace: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5265–5274.

[10] W.-F. Ou, L.-M. Po, C. Zhou, Y.-J. Zhang, L.-T. Feng, Y. A. U. Rehman, and Y.-Z. Zhao, "LinCos-Softmax: Learning Angle-Discriminative Face Representations with Linearity-Enhanced Cosine Logits," *IEEE Access*, vol. 8, pp. 109 758–109 769, 2020.

[11] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition*, 2008.

[12] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient cnns for accurate real-time face verification on mobile devices," in *Chinese Conference on Biometric Recognition*, 2018, pp. 428–438.