Report

Team: It's FaceTime

Patil Bhavesh Vasant | 18D170023

Anuj Agrawal | 180110012

Abstract

A general statement of the problem of machine recognition of faces can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces. Face recognition is often described as a process that first involves four steps; they are: face detection, face alignment, feature extraction, and finally face recognition. The human face is a dynamic object and has a high degree of variability in its appearance, which makes face detection a difficult problem in computer vision*.*

It can be further analysed as

- **Face Verification**. A one-to-one mapping of a given face against a known identity (e.g. *is this the person?*).
- **Face Identification**. A one-to-many mapping for a given face against a database of known faces (e.g. *who is this person?*)

We use a unique **ArcFace** [1] based approach for calculation of face embeddings for the faces present in database. ***ArcFace* is a machine learning model that takes two face images as input and outputs the distance between them to see how likely they are to be the same person. It can be used for face recognition and face search**.

We use train.csv to do comparative analysis of cosine scores for matching and non-matching face embeddings calculated with ArcFace.

Method

ArcFace uses a *similarity learning* mechanism that allows *distance metric learning* to be solved in the classification task by introducing *Angular Margin Loss* to replace *Softmax Loss.*

The distance between faces is calculated using *cosine distance*, which is a method used by search engines and can be calculated by the inner product of two normalized vectors. If the two vectors are the same, θ will be 0 and cosθ=1. If they are orthogonal, θ will be π/2 and cosθ=0. Therefore, it can be used as a similarity measure.

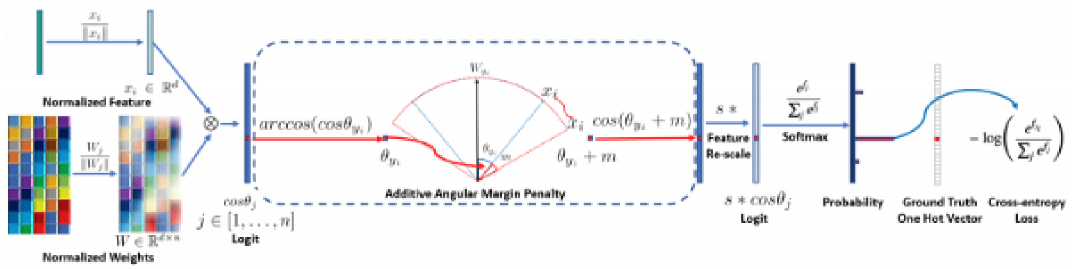


Figure 2. Training a DCNN for face recognition supervised by the ArcFace loss. Based on the feature x_i and weight W normalisation, we get the $\cos\theta_j$ (logit) for each class as $W_j^T x_i$. We calculate the $arccos\theta_{y_i}$ and get the angle between the feature x_i and the ground truth weight W_{y_i}. In fact, W_j provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θ_{y_i}. After that, we calculate $\cos(\theta_{y_i} + m)$ and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross entropy loss.

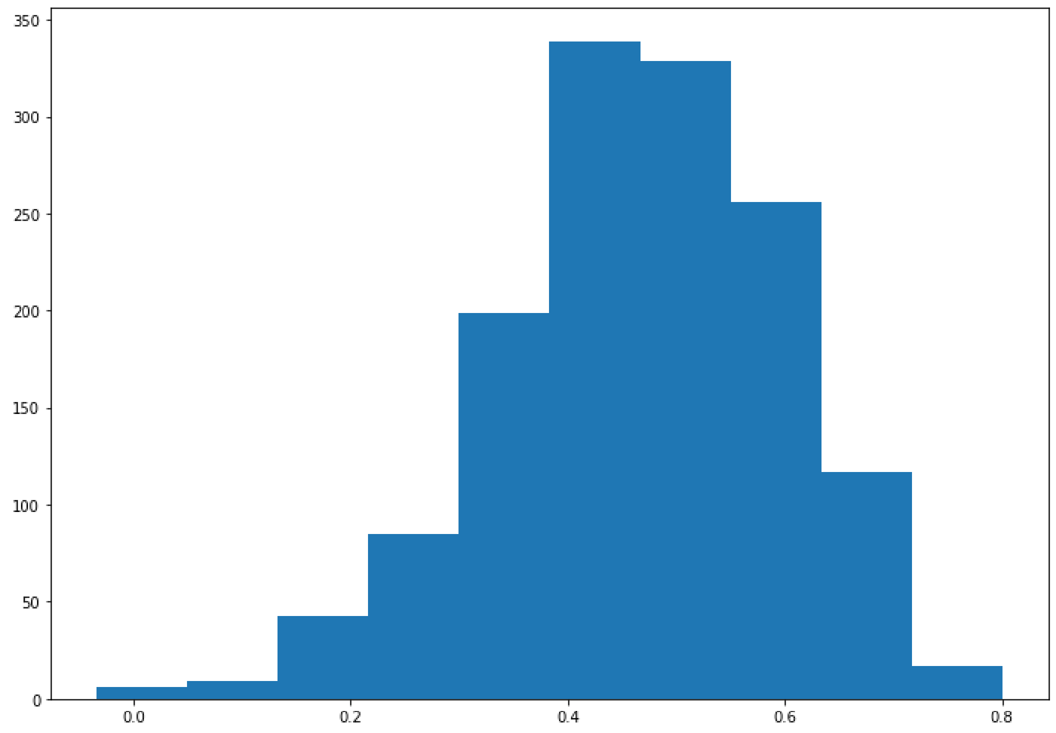
Algorithm 1 The Pseudo-code of ArcFace on MxNet

```
Input: Feature Scale $s$, Margin Parameter $m$ in Eq. 3, Class Number $n$, Ground-Truth ID $gt$.
  1. x = mx.symbol.L2Normalization (x, mode = 'instance')
  2. W = mx.symbol.L2Normalization (W, mode = 'instance')
  3. fc7 = mx.sym.FullyConnected (data = x, weight = W, no_bias = True, num_hidden = n)
  4. original_target_logit = mx.sym.pick (fc7, gt, axis = 1)
  5. theta = mx.sym.arccos (original_target_logit)
  6. marginal_target_logit = mx.sym.cos (theta + m)
  7. one_hot = mx.sym.one_hot (gt, depth = n, on_value = 1.0, off_value = 0.0)
  8. fc7 = fc7 + mx.sym.broadcast_mul (one_hot, mx.sym.expand_dims (marginal_target_logit - original_target_logit, 1))
  9. fc7 = fc7 * s
Output: Class-wise affinity score $fc7$.
```

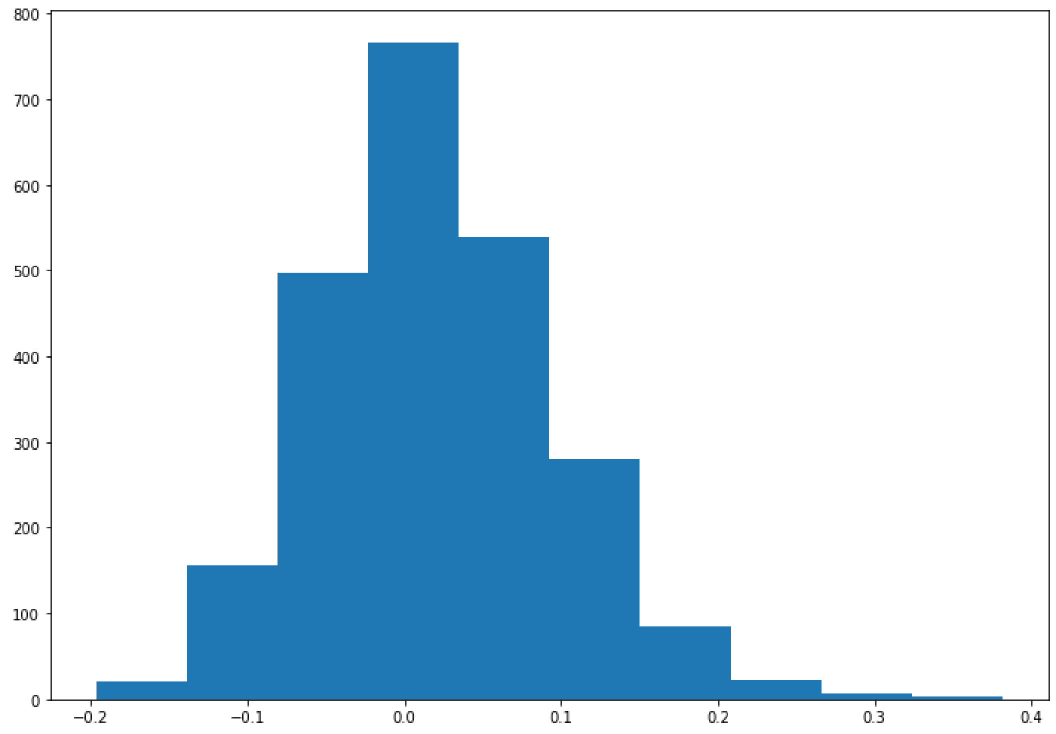
We use **insightface** [2] open source implementation for arcface which calculates the face embeddings(size: vector of 512 direction) for the faces. We used no particular accelerator but CPU for this approach. We used google colab and mounted google drive to fetch the data from the shared drive.

We took first 5000 datapoints into consideration to take inferences. We calculated cosine score for matching face embeddings and non-matching face embeddings and observed that the optimal threshold for the given facial data distribution lier around **0.2.**

Histogram for matching face embeddings



Histogram for non-matching face embeddings



Note: X-axis: cosine score; Y-axis: Number of datapoints

Hence from the above observations cosine score of 0.2 is taken as a threshold for differentiating the matching and non-matching face embeddings.

💡 Cosine Score ≥ 0.2 —> Faces are matching and belong to same person

💡 Cosine Score < 0.2 —> Faces are matching and belong to different persons

Code

> https://github.com/freaky-perceptron/FaceMatch_challenge

References

[1] ArcFace: Additive Angular Margin Loss for Deep Face Recognition; https://arxiv.org/pdf/1801.07698.pdf

[2] InsightFace: 2D and 3D Face Analysis Project; https://github.com/deepinsight/insightface

[3] Deep Face Recognition: A Survey; https://arxiv.org/pdf/1804.06655.pdf

[4] FaceNet: A Unified Embedding for Face Recognition and Clustering; https://arxiv.org/pdf/1503.03832.pdf